Mini Project report

On

**CLOUD CONTROLLED ROBOT**

Submitted in partial fulfillment of the requirement for the award of degree of

**BACHELOR OF TECHNOLOGY**

in

**COMPUTER SCIENCE & ENGINEERING**

Submitted by

**Akash Raman (13311A05E4)**

**Pranav Reddy (13311A05E7)**

**Jatin Pratap (13311A05G5)**

Under the Guidance of

K.Sirisha

Associate Professor

Department of CSE



**Department of Computer Science & Engineering**

**SREENIDHI INSTITUTE OF SCIENCE AND TECHNOLOGY**

**(Affiliated to Jawaharlal Nehru Technological University,Hyderabad)**

Yamnampet (V), Ghatkesar (M), Hyderabad – 501301,Telangana.

**2016-2017**

**Department Of Computer Science & Engineering**

**SREENIDHI INSTITUTE OF SCIENCE AND TECHNOLOGY**

**(Affiliated to Jawaharlal Nehru Technological University, Hyderabad)**

Yamnampet (V), Ghatkesar (M), Hyderabad – 501301, Telangana.



CERTIFICATE

This is to certify that the project entitled “**CLOUD CONTROLLED ROBOT**” is being submitted by

**Akash Raman (13311A05E4)**

**Pranav Reddy (13311A05E7)**

**Jatin Pratap (13311A05G5)**

in partial fulfillment of the requirements for the award of **BACHELOR OF TECHNOLOGY** to **JNTU, Hyderabad**. This record is a bonafide work carried out by them under my guidance and supervision. The result embodied in this project report has not been submitted to any other university or institute for the award of any degree of diploma.

**Internal guide Project Coordinator Head of Department**

K.Sirisha K.C.Sreedhar Dr. ARUNA VARNASI

Associate Professor Assistant Professor Professor& H.O.D

Department of CSE Department of CSE Department of CSE

**External Examiner**

**LIST OF CONTENTS**

**Description**  **Page**

**No.**

1. INTRODUCTION **1**

1.1 Purpose 1

1.2 Scope 1

1.3 System Architecture 2

1.4 Overview 3

2. LITERATURE SURVEY **4**

2.1 About Arduino

2.1.1 Hardware 4

2.1.2 Software 5

2.2 Java Script 7

2.1.1 Java Script Statements 7

2.1.2 Java Script Versus Java 8

2.3Hyper text markup language 9

2.4 Android 12

2.5 Servlets 17

3. SYSTEM ANALYSIS **21**

3.1 Existing System 21

3.1.1Disadvantages 21

3.2 Proposed System 22

3.2.1 Advantages 22

4. SYSTEM REQUIREMENTS SPECIFICATION **23**

4.1 Hardware Requirements 23

4.2 Software Requirements 23

5. SYSTEM DESIGN **24**

5.1 System Specification 24

5.2 System Components 24

5.3 Uml Diagrams 25

5.4 Data Dictionary

7. SYSTEM TESTING  **28**

7.1 Code Testing 28

7.2 Specification Testing 29

7.3 Levels of Testing 30

7.3.1 Unit Testing 30

7.3.2 System Testing 30

7.4 Sample Tests **31**

8. CONCLUSION AND FUTURE ENHANCEMENTS **32**

9. REFERENCES **33**

**1.INTRODUCTION**

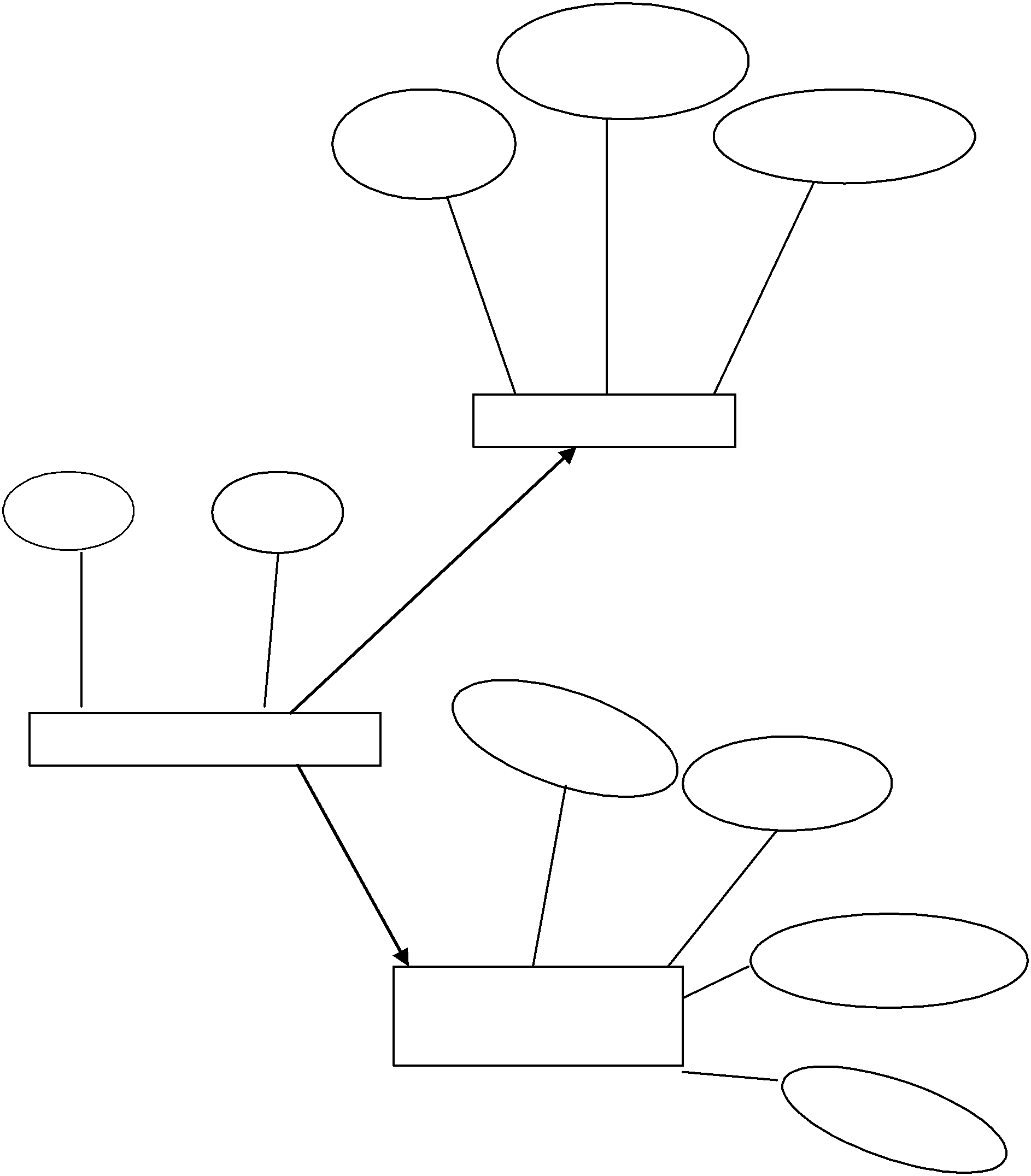
**1.1.PURPOSE**

The purpose of this document is to display all the external requirements for the “Cloud Controlled Robot”. The main objective of preparing this document is to give a detailed description of the analysis and requirements for the system to be automated and this will be a guide in the other phase. The purpose of “Cloud Controlled Robot “ is to control the robot from anywhere around the world, Provided with Internet.

**1.2 SCOPE**

This document is meant for use by the developers and will be the basis for developing the final Robot. The requirement changes to the document can be made due to changes in the client needs, change in technology, addition of more sensors, functions etc.,

**1.3 SYSTEM ARCHITECTURE**



**1.4 OVERVIEW OF THE PROJECT**

# Cloud Controlled Robot is equipped with ESP8266.Which is a Wi-Fi enabled module that controls robot from any terrain around the world. All the commands are stored on this chip using the Arduino software. The commands are embedded in the chip by using the software provided with the connection to PC via USB the code is dumped into the Wi-Fi enabled module

# The Robot can be controlled by phone or webpage interface through which the cloud commands which are stored are the triggered . those commands are then retrieved by ESP8266 and then the robot will start functioning. We can move the robot forward, backward and sideways by pressing their respective allotted keys on the smart phone or the web page.

# Thus the power to control the robot from any region makes the robot very useful in every field. The robot can be tracked easily and new functions and features can easily be added thus making a lot of scope for further improvisation to the robot

User Interface Requirements : Web Based, User friendly interface

Integration Requirement : Web Enabled, Well integrated with internal system

Preferred Technologies : Arduino IDE : Arduino/C/HTML/CSS/Java Script/Blynk.

**2. LITERATURE SURVEY**

**2.1 ABOUT ARDUINO**

**Arduino** is an open-source project that created microcontroller-based kits for building digital devices and interactive objects that can sense and control physical devices.

The project is based on microcontroller board designs, produced by several vendors, using various microcontrollers. These systems provide sets of digital and analog input/output (I/O) pins that can interface to various expansion boards (termed *shields*) and other circuits. The boards feature serial communication interfaces, including Universal Serial Bus (USB) on some models, for loading programs from personal computers. For programming the microcontrollers, the Arduino project provides an integrated development environment (IDE) based on a programming language named *Processing*, which also supports the languages [C](https://en.wikipedia.org/wiki/C_(programming_language)) and C++.

The first Arduino was introduced in 2005, aiming to provide a low cost, easy way for novices and professionals to create devices that interact with their environment using sensors and actuators. Common examples of such devices intended for beginner hobbyists include simple robots, thermostats, and motion detectors.

Arduino boards are available commercially in preassembled form, or as do-it-yourself kits. The hardware design specifications are openly available, allowing the Arduino boards to be produced by anyone. Adafruit Industries estimated in mid-2011 that over 300,000 official Arduinos had been commercially produced,[[2]](https://en.wikipedia.org/wiki/Arduino#cite_note-2) and in 2013 that 700,000 official boards were in users' hands.[[3]](https://en.wikipedia.org/wiki/Arduino#cite_note-3)

**2.1.1 HARDWARE**

An Arduino board consists of an Atmel 8-, 16- or 32-bit AVR microcontroller (although since 2015 other makers' microcontrollers have been used) with complementary components that facilitate programming and incorporation into other circuits. An important aspect of the Arduino is its standard connectors, which let users connect the CPU board to a variety of interchangeable add-on modules termed *shields*. Some shields communicate with the Arduino board directly over various pins, but many shields are individually addressable via an I²C serial bus—so many shields can be stacked and used in parallel. Before 2015, Official Arduinos had used the Atmel megaAVR series of chips, specifically the ATmega8, ATmega168, ATmega328, ATmega1280, and [ATmega2560](https://en.wikipedia.org/w/index.php?title=ATmega2560&action=edit&redlink=1). In 2015, units by other producers were added. A handful of other processors have also been used by Arduino compatible devices. Most boards include a 5 V linear regulator and a 16 MHz crystal oscillator (or ceramic resonator in some variants), although some designs such as the LilyPad run at 8 MHz and dispense with the onboard voltage regulator due to specific form-factor restrictions. An Arduino's microcontroller is also pre-programmed with a boot loader that simplifies uploading of programs to the on-chip flash memory, compared with other devices that typically need an external chip programmer. This makes using an Arduino more straightforward by allowing the use of an ordinary computer as the programmer. Currently, optiboot bootloader is the default bootloader installed on Arduino UNO.[[5]](https://en.wikipedia.org/wiki/Arduino#cite_note-5)

At a conceptual level, when using the Arduino integrated development environment, all boards are programmed over a serial connection. Its implementation varies with the hardware version. Some serial Arduino boards contain a level shifter circuit to convert between RS-232 logic levels and transistor–transistor logic (TTL) level signals. Current Arduino boards are programmed via Universal Serial Bus (USB), implemented using USB-to-serial adapter chips such as the [FTDI](https://en.wikipedia.org/wiki/FTDI) FT232. Some boards, such as later-model Uno boards, substitute the FTDI chip with a separate AVR chip containing USB-to-serial firmware, which is reprogrammable via its own ICSP header. Other variants, such as the Arduino Mini and the unofficial Boarduino, use a detachable USB-to-serial adapter board or cable, Bluetooth or other methods, when used with traditional microcontroller tools instead of the Arduino IDE, standard AVR in-system programming (ISP) programming is used.

The Arduino board exposes most of the microcontroller's I/O pins for use by other circuits. The *Diecimila*,[[a]](https://en.wikipedia.org/wiki/Arduino#cite_note-N10000-6) *Duemilanove*,[[b]](https://en.wikipedia.org/wiki/Arduino#cite_note-N2009-7) and current *Uno*[[c]](https://en.wikipedia.org/wiki/Arduino#cite_note-N1-8) provide 14 digital I/O pins, six of which can produce pulse-width modulated signals, and six analog inputs, which can also be used as six digital I/O pins. These pins are on the top of the board, via female 0.1-inch (2.54 mm) headers. Several plug-in application shields are also commercially available. The Arduino Nano, and Arduino-compatible Bare Bones Board[[6]](https://en.wikipedia.org/wiki/Arduino#cite_note-9) and Boarduino[[7]](https://en.wikipedia.org/wiki/Arduino" \l "cite_note-10) boards may provide male header pins on the underside of the board that can plug into solderless breadboards.

Many Arduino-compatible and Arduino-derived boards exist. Some are functionally equivalent to an Arduino and can be used interchangeably. Many enhance the basic Arduino by adding output drivers, often for use in school-level education, to simplify making buggies and small robots. Others are electrically equivalent but change the form factor, sometimes retaining compatibility with shields, sometimes not. Some variants use different processors, of varying compatibility.**]**

**2.1.2 SOFTWARE**

The Arduino project provides the Arduino integrated development environment (IDE), which is a cross-platform application written in the programming language [Java](https://en.wikipedia.org/wiki/Java_(programming_language)). It originated from the IDE for the languages *Processing* and *Wiring*. It is designed to introduce programming to artists and other newcomers unfamiliar with software development. It includes a code editor with features such as syntax highlighting, brace matching, and automatic indentation, and provides simple one-click mechanism to compile and load programs to an Arduino board. A program written with the IDE for Arduino is called a "sketch".[[15]](https://en.wikipedia.org/wiki/Arduino#cite_note-19)

The Arduino IDE supports the languages [C](https://en.wikipedia.org/wiki/C_(programming_language)) and [C++](https://en.wikipedia.org/wiki/C%2B%2B) using special rules to organize code. The Arduino IDE supplies a software library called Wiring from the Wiring project, which provides many common input and output procedures. A typical Arduino C/C++ sketch consist of two functions that are compiled and linked with a program stub *main()* into an executable cyclic executive program:

* *setup()*: a function that runs once at the start of a program and that can initialize settings.
* *loop()*: a function called repeatedly until the board powers off.

After compiling and linking with the [GNU toolchain](https://en.wikipedia.org/wiki/GNU_toolchain), also included with the IDE distribution, the Arduino IDE employs the program *avrdude* to convert the executable code into a text file in hexadecimal coding that is loaded into the Arduino board by a loader program in the board's firmware.

## 2.2 JavaScript

JavaScript is a script-based programming language that was developed by Netscape Communication Corporation. JavaScript was originally called Live Script and renamed as JavaScript to indicate its relationship with Java. JavaScript supports the development of both client and server components of Web-based applications. On the client side, it can be used to write programs that are executed by a Web browser within the context of a Web page. On the server side, it can be used to write Web server programs that can process information submitted by a Web browser and then updates the browser’s display accordingly Even though JavaScript supports both client and server Web programming, we prefer JavaScript at Client side programming since most of the browsers supports it. JavaScript is almost as easy to learn as HTML, and JavaScript statements can be included in HTML documents by enclosing the statements between a pair of scripting tags

**<SCRIPTS>...</SCRIPT>.**

**<SCRIPT LANGUAGE = “JavaScript”>;**

**2.2.1 JAVASCRIPT STATEMENTS**

**</SCRIPT>**

Here are a few things we can do with JavaScript :

* Validate the contents of a form and make calculations.
* Add scrolling or changing messages to the Browser’s status line
* Animate images or rotate images that change when we move the mouse over them.
* Detect the browser in use and display different content for different browsers.
* Detect installed plug-ins and notify the user if a plug-in is required.
* We can do much more with JavaScript, including creating entire application.

**2.2.2 JAVASCRIPT VERSUS JAVA**

JavaScript and Java are entirely different languages. A few of the most glaring differences are:

* Java applets are generally displayed in a box within the web document; JavaScript can affect any part of the Web document itself.
* While JavaScript is best suited o simple applications and adding interactive features to Web pages; Java can be used for incredibly complex applications.
* There are many other differences but the important thing to remember is that JavaScript and Java are separate languages. They are both useful for different things; in fact they can be used together to combine their advantages.

**2.2.3 ADVANTAGES**

* JavaScript can be used for Sever-side and Client-side scripting.
* It is more flexible than VBScript.
* JavaScript is the default scripting languages at Client-side since all the browsers supports it.

**2.3 HYPERTEXT MARKUP LANGUAGE**

Hypertext Markup Language (HTML), the languages of the World Wide Web (WWW), allows users to produces Web pages that include text, graphics and pointer to otherWeb pages (Hyperlinks). HTML is not a programming language but it is an application of ISO Standard 8879,SGML (Standard Generalized Markup Language), but specialized to hypertext and adapted to the Web. The idea behind Hypertext is that instead of reading text in rigid linear structure, we can easily jump from one point to another point. We can navigate through the information based on our interest and preference. A markup language is simply a series of elements, each delimited with special characters that define how text or other items enclosed within the elements should be displayed. Hyperlinks are underlined or emphasized works that load to other documents or some portions of the same document. HTML can be used to display any type of document on the host computer, which can be geographically at a different location. It is a versatile language and can be used on any platform or desktop. HTML provides tags (special codes) to make the document look attractive. HTML tags are not case-sensitive. Using graphics, fonts, different sizes, color, etc., can enhance the presentation of the document. Anything that is not a tag is part of the document itself.

**2.3.1 BASIC HTML TAGS**

<!-- --> Specifies comments

<A>……….</A>; Creates hypertext links

<B>……….</B>; Formats text as bold

<BIG>……….</BIG>; Formats text in large font.

<BODY>;…</BODY>; Contains all tags and text in the HTML document

<CENTER>...</CENTER> Creates text

<DD>…</DD> Definition of a term

<DL>...</DL> Creates definition list

<FONT>;…</FONT> Formats text with a particular font

<FORM>...</FORM> Encloses a fill-out form

<FRAME>;...</FRAME> Defines a particular frame in a set of frames

<H#>…</H#> Creates headings of different levels

<HEAD>...</HEAD> Contains tags that specify information about a document

<HR>...<HR> Creates a horizontal rule

<HTML>…</HTML> Contains all other HTML tags

<META>...</META> Provides meta-information about a document

<SCRIPT>;…</SCRIPT>; Contains client-side or server-side script

<TABLE>…</TABLE>; Creates a table

<TD>…</TD> Indicates table data in a table

<TR>…</TR> Designates a table row

<TH>…</TH>; Creates a heading in a table

**2.3.2 ADVANTAGES**

A HTML document is small and hence easy to send over the net. It is small because it does not include formatted information. HTML is platform independent. HTML tags are not case-sensitive.

**2.4 ABOUT ANDROID**

Android apps are written in the Java programming language. The Android SDK tools compile your code—along with any data and resource files—into an APK: an *Android package*, which is an archive file with an .apk suffix. One APK file contains all the contents of an Android app and is the file that Android-powered devices use to install the app.

Once installed on a device, each Android app lives in its own security sandbox:

* The Android operating system is a multi-user Linux system in which each app is a different user.
* By default, the system assigns each app a unique Linux user ID (the ID is used only by the system and is unknown to the app). The system sets permissions for all the files in an app so that only the user ID assigned to that app can access them.
* Each process has its own virtual machine (VM), so an app's code runs in isolation from other apps.
* By default, every app runs in its own Linux process. Android starts the process when any of the app's components need to be executed, then shuts down the process when it's no longer needed or when the system must recover memory for other apps.

In this way, the Android system implements the principle of least privilege. That is, each app, by default, has access only to the components that it requires to do its work and no more. This creates a very secure environment in which an app cannot access parts of the system for which it is not given permission.

However, there are ways for an app to share data with other apps and for an app to access system services:

* It's possible to arrange for two apps to share the same Linux user ID, in which case they are able to access each other's files. To conserve system resources, apps with the same user ID can also arrange to run in the same Linux process and share the same VM (the apps must also be signed with the same certificate).
* An app can request permission to access device data such as the user's contacts, SMS messages, the mountable storage (SD card), camera, Bluetooth, and more. The user has to explicitly grant these permissions. For more information, see [Working with System Permissions](https://developer.android.com/training/permissions/index.html).

That covers the basics regarding how an Android app exists within the system. The rest of this document introduces you to:

* The core framework components that define your app.
* The manifest file in which you declare components and required device features for your app.
* Resources that are separate from the app code and allow your app to gracefully optimize its behavior for a variety of device configurations.

**2.4.1 APP COMPONENTS**

App components are the essential building blocks of an Android app. Each component is a different point through which the system can enter your app. Not all components are actual entry points for the user and some depend on each other, but each one exists as its own entity and plays a specific role—each one is a unique building block that helps define your app's overall behavior.

There are four different types of app components. Each type serves a distinct purpose and has a distinct lifecycle that defines how the component is created and destroyed.

Here are the four types of app components:

**Activities**

An *activity* represents a single screen with a user interface. For example, an email app might have one activity that shows a list of new emails, another activity to compose an email, and another activity for reading emails. Although the activities work together to form a cohesive user experience in the email app, each one is independent of the others. As such, a different app can start any one of these activities (if the email app allows it). For example, a camera app can start the activity in the email app that composes new mail, in order for the user to share a picture.

An activity is implemented as a subclass of [Activity](https://developer.android.com/reference/android/app/Activity.html) and you can learn more about it in the [Activities](https://developer.android.com/guide/components/activities.html) developer guide.

**Services**

A *service* is a component that runs in the background to perform long-running operations or to perform work for remote processes. A service does not provide a user interface. For example, a service might play music in the background while the user is in a different app, or it might fetch data over the network without blocking user interaction with an activity. Another component, such as an activity, can start the service and let it run or bind to it in order to interact with it.

A service is implemented as a subclass of [Service](https://developer.android.com/reference/android/app/Service.html) and you can learn more about it in the [Services](https://developer.android.com/guide/components/services.html) developer guide.

**Content providers**

A *content provider* manages a shared set of app data. You can store the data in the file system, an SQLite database, on the web, or any other persistent storage location your app can access. Through the content provider, other apps can query or even modify the data (if the content provider allows it). For example, the Android system provides a content provider that manages the user's contact information. As such, any app with the proper permissions can query part of the content provider (such as [ContactsContract.Data](https://developer.android.com/reference/android/provider/ContactsContract.Data.html)) to read and write information about a particular person.

Content providers are also useful for reading and writing data that is private to your app and not shared. For example, the [Note Pad](https://developer.android.com/resources/samples/NotePad/index.html) sample app uses a content provider to save notes.

A content provider is implemented as a subclass of [ContentProvider](https://developer.android.com/reference/android/content/ContentProvider.html) and must implement a standard set of APIs that enable other apps to perform transactions. For more information, see the [Content Providers](https://developer.android.com/guide/topics/providers/content-providers.html) developer guide.

**Broadcast receivers**

A *broadcast receiver* is a component that responds to system-wide broadcast announcements. Many broadcasts originate from the system—for example, a broadcast announcing that the screen has turned off, the battery is low, or a picture was captured. Apps can also initiate broadcasts—for example, to let other apps know that some data has been downloaded to the device and is available for them to use. Although broadcast receivers don't display a user interface, they may [create a status bar notification](https://developer.android.com/guide/topics/ui/notifiers/notifications.html) to alert the user when a broadcast event occurs. More commonly, though, a broadcast receiver is just a "gateway" to other components and is intended to do a very minimal amount of work. For instance, it might initiate a service to perform some work based on the event.

A broadcast receiver is implemented as a subclass of [BroadcastReceiver](https://developer.android.com/reference/android/content/BroadcastReceiver.html) and each broadcast is delivered as an [Intent](https://developer.android.com/reference/android/content/Intent.html) object. For more information, see the [BroadcastReceiver](https://developer.android.com/reference/android/content/BroadcastReceiver.html) class.

A unique aspect of the Android system design is that any app can start another app’s component. For example, if you want the user to capture a photo with the device camera, there's probably another app that does that and your app can use it, instead of developing an activity to capture a photo yourself. You don't need to incorporate or even link to the code from the camera app. Instead, you can simply start the activity in the camera app that captures a photo. When complete, the photo is even returned to your app so you can use it. To the user, it seems as if the camera is actually a part of your app.

When the system starts a component, it starts the process for that app (if it's not already running) and instantiates the classes needed for the component. For example, if your app starts the activity in the camera app that captures a photo, that activity runs in the process that belongs to the camera app, not in your app's process. Therefore, unlike apps on most other systems, Android apps don't have a single entry point (there's no main()function, for example).

Because the system runs each app in a separate process with file permissions that restrict access to other apps, your app cannot directly activate a component from another app. The Android system, however, can. So, to activate a component in another app, you must deliver a message to the system that specifies your intent to start a particular component. The system then activates the component for you.

### 2.4.2 Activating Components

Three of the four component types—activities, services, and broadcast receivers—are activated by an asynchronous message called an intent. Intents bind individual components to each other at runtime (you can think of them as the messengers that request an action from other components), whether the component belongs to your app or another.

An intent is created with an [Intent](https://developer.android.com/reference/android/content/Intent.html) object, which defines a message to activate either a specific component or a specific type of component—an intent can be either explicit or implicit, respectively.

For activities and services, an intent defines the action to perform (for example, to "view" or "send" something) and may specify the URI of the data to act on (among other things that the component being started might need to know). For example, an intent might convey a request for an activity to show an image or to open a web page. In some cases, you can start an activity to receive a result, in which case, the activity also returns the result in an [Intent](https://developer.android.com/reference/android/content/Intent.html) (for example, you can issue an intent to let the user pick a personal contact and have it returned to you—the return intent includes a URI pointing to the chosen contact).

For broadcast receivers, the intent simply defines the announcement being broadcast (for example, a broadcast to indicate the device battery is low includes only a known action string that indicates "battery is low").

The other component type, content provider, is not activated by intents. Rather, it is activated when targeted by a request from a [ContentResolver](https://developer.android.com/reference/android/content/ContentResolver.html). The content resolver handles all direct transactions with the content provider so that the component that's performing transactions with the provider doesn't need to and instead calls methods on the [ContentResolver](https://developer.android.com/reference/android/content/ContentResolver.html) object. This leaves a layer of abstraction between the content provider and the component requesting information (for security).

There are separate methods for activating each type of component:

You can start an activity (or give it something new to do) by passing an [Intent](https://developer.android.com/reference/android/content/Intent.html) to [startActivity()](https://developer.android.com/reference/android/content/Context.html" \l "startActivity(android.content.Intent)) or [startActivityForResult()](https://developer.android.com/reference/android/app/Activity.html" \l "startActivityForResult(android.content.Intent, int)) (when you want the activity to return a result).

You can start a service (or give new instructions to an ongoing service) by passing an [Intent](https://developer.android.com/reference/android/content/Intent.html) to [startService()](https://developer.android.com/reference/android/content/Context.html" \l "startService(android.content.Intent)). Or you can bind to the service by passing an [Intent](https://developer.android.com/reference/android/content/Intent.html) to [bindService()](https://developer.android.com/reference/android/content/Context.html" \l "bindService(android.content.Intent, android.content.ServiceConnection, int)).

You can initiate a broadcast by passing an [Intent](https://developer.android.com/reference/android/content/Intent.html) to methods like [sendBroadcast()](https://developer.android.com/reference/android/content/Context.html" \l "sendBroadcast(android.content.Intent)), [sendOrderedBroadcast()](https://developer.android.com/reference/android/content/Context.html" \l "sendOrderedBroadcast(android.content.Intent, java.lang.String)), or [sendStickyBroadcast()](https://developer.android.com/reference/android/content/Context.html" \l "sendStickyBroadcast(android.content.Intent)).

You can perform a query to a content provider by calling [query()](https://developer.android.com/reference/android/content/ContentProvider.html#query(android.net.Uri, java.lang.String[], java.lang.String, java.lang.String[], java.lang.String)) on a [ContentResolver](https://developer.android.com/reference/android/content/ContentResolver.html).

For more information about using intents, see the [Intents and Intent Filters](https://developer.android.com/guide/components/intents-filters.html) document. More information about activating specific components is also provided in the following documents: [Activities](https://developer.android.com/guide/components/activities.html), [Services](https://developer.android.com/guide/components/services.html), [BroadcastReceiver](https://developer.android.com/reference/android/content/BroadcastReceiver.html) and [Content Providers](https://developer.android.com/guide/topics/providers/content-providers.html).

**2.5 SERVLETS**

**INTRODUCTION**

The Java web server is JavaSoft&#39;s own web Server. The Java web server is just a part of a larger framework, intended to provide you not just with a web server, but also with tools. To build customized network servers for any Internet or Intranet client/server system. Servlets are to a web server, how applets are to the browser.

**ABOUT SERVLETS**

Servlets provide a Java-based solution used to address the problems currently associated with doing server-side programming, including inextensible scripting solutions, platform-specific APIs, and incomplete interfaces. Servlets are objects that conform to a specific interface that can be plugged into a Java-based server. Servlets are to the server-side what applets are to the client-side – object byte codes that can be dynamically loaded off the net. They differ from applets in that they are faceless objects (without graphics or a GUI component). They serve as platform independent, dynamically loadable, plugable helper byte code objects on the server side that can be used to dynamically extend server-side functionality.

For example, an HTTP Servlets can be used to generate dynamic HTML content. When you use Servlets to do dynamic content you get the following advantages:

* They’re faster and cleaner than CGI scripts
* They use a standard API (the Servlets API)
* They provide all the advantages of Java (run on a variety of servers without needing to be rewritten).

**ATTRACTIVENESS OF SERVLETS**

There are many features of Servlets that make them easy and attractive to use. These include:

Easily configured using the GUI-based Admin tool

* Can be loaded and invoked from a local disk or remotely across the network.
* Can be linked together, or chained, so that one Servlets can call another Servlets, or several Servlets in sequence.
* Can be called dynamically from within HTML pages, using server-side include tags. Are secure - even when downloading across the network, the Servlets security model and Servlets sandbox protect your system from unfriendly behavior.

**ADVANTAGES OF SERVLET API**

One of the great advantages of the Servlet API is protocol independence. It assumes nothing about:

* The protocol being used to transmit on the net
* How it is loaded
* The server environment it will be running in

These qualities are important, because it allows the Servlet API to be embedded in many different kinds of servers. There are other advantages to the Servlet API as well. These

include:

* It’s extensible - you can inherit all your functionality from the base classes made available to you.
* it&#39;s simple, small, and easy to use.

**FEATURES OF SERVLETS**

* Servlets are fast. Since Servlets only need to be loaded once, they offer much better performance over their CGI counterparts.
* Servlets are platform independent.
* Servlets are extensible. Java is a robust, object-oriented programming language, which easily can be extended to suit your needs
* Servlets are secure.
* Servlets can be used with a variety of clients.

**LOADING SERVLETS**

Servlets can be loaded from three places From a directory that is on the CLASSPATH. The CLASSPATH of the JavaWebServer includes service root/classes/ which is where the system classes reside.From the &lt;SERVICE\_ROOT /Servlets/ directory. This is \*not\* in the server&#39;s classpath. A class loader is used to create Servlets from this directory. New Servlets can be added - existing Servlets can be recompiled and the server will notice these changes. From a remote location. For this a code base like http: // nine.eng / classes / foo / is required in addition to the Servlets class name. Refer to the admin GUI docs on Servletsection to see how to set this up.

**LOADING REMOTE SERVLETS**

Remote Servlets can be loaded by:

1. Configuring the Admin Tool to setup automatic loading of remote Servlets

2. Setting up server side include tags in .shtml files

3. Defining a filter chain configuration

**INVOKING SERVLETS**

A Servlet invoker is a Servlet that invokes the &quot;service&quot; method on a named Servlet. If the Servlet is not loaded in the server, then the invoker first loads the Servlet (either from local disk or from the network) and the then invokes the &quot;service&quot; method. Also like applets, local Servlets in the server can be identified by just the class name. In other words, if a Servlet name is not absolute, it is treated as local. A client can invoke Servlets in the following ways:

* The client can ask for a document that is served by the Servlet.
* The client (browser) can invoke the Servlet directly using a URL, once it has been mapped using the Servlet Aliases section of the admin GUI.
* The Servlet can be invoked through server side include tags.
* The Servlet can be invoked by placing it in the Servlets/ directory.
* The Servlet can be invoked by using it in a filter chain.

**3. SYSTEM ANALYSIS**

**3.1 Definition and reason for Condition Analysis**

System analysis will be performed to determine if it is feasible to design an information based on policies and plans of the organization and on user requirements and to eliminate the weaknesses of the present system.

General requirements

1. The new system should be cost effective.

2. To augment management, improve productivity and services.

3. To enhance User/System interface.

4. To improve information qualify and usability.

5. To upgrade system’s reliability, availability, flexibility and growth potential.

**3.2 EXISTING SYSTEM**

**3.2.1 DISADVANTAGES:**

* In the existing system the robot is not based on cloud commuting it uses the Bluetooth module and thus making is accessible to limited region only.
* The cloud model has a time delay and thus making the robot slow and there is delay in the moment of the robot

**3.3 PROPOSED SYSTEM**

* The cloud based robot is much faster is uses blynk cloud platform that is makes much faster to run the commands and it can controlled from any region from around the world.

**4. REQUIREMENT SPECIFICIATIONS**

**4.1 SOFTWARE REQUIREMENT SPECIFICIATIONS**

* Language (front end) : embedded C
* Web technology :HTML,Java Sciprt,Css.
* Arduino IDE : Arduino
* Cloud platform : Blynk

**4.2 HARDWARE REQUIREMENTS SPECIFICATIONS**

* esp8266
* L298N Motor-driver PCB or Ardumoto shield
* Bread board
* 2x GM9 geared motors
* 2x GMPV wheels
* wires

**5.SYSTEM DESIGN**

**5.1 SYSTEM SPECIFICATIONS**

System specification documents most predominantly contain information on basic website requirements which include:

Performance levels

Reliability

Quality

Interfaces

Security and Privacy

Constraints and Limitations

Functional Capabilities

Data Structures and elements

System specifications are:

1. Pentium 4 with minimum 1.x GHz processor or equivalent processor

2. Minimum 128 MB RAM (1 GB RAM recommended)

3. Hard disk with minimum 1 GB free space

4. NIC (network interface card) connected to network pentium III

5. Ram : 64 MB.

6. Hard disk : 10.2 GB.

7. Monitor : SVGA color monitor.

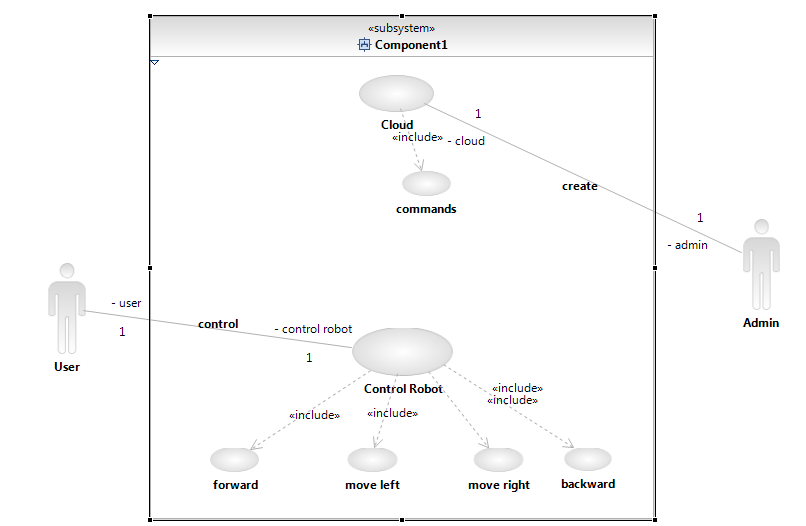
8. Keyboard : 105 standard mouse.

**5.3 UML DIAGRAMS**

**5.3.1 USE CASE DIAGRAM:**

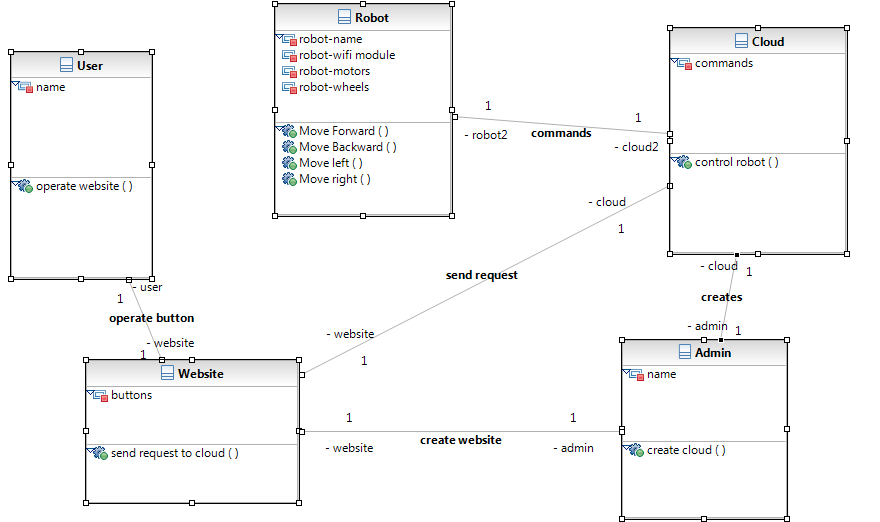
A use case is a set of scenarios that describing an interaction between a user and a system. A use case diagram displays the relationship among actors and use cases. The two main components of a use case diagram are use cases and actors. An actor is represents a user or another system that will interact with the system you are modeling. A use case is an external view of the system that represents some action the usermight perform in order to complete a task.

They are helpful in exposing requirements and planning the project..



**5.3.2 CLASS DIAGRAM**

In software engineering, a **class diagram** in the Unified Modeling Language (UML) is a type of static structure **diagram** that describes the structure of a system by showing the system's **classes**, their attributes, operations (or methods), and the relationships among objects.



**5.4 DATA DICTIONARY**

The logical characteristics of current systems data stores, including name, description,aliases, contents, and organization, identifies processes where the data are used and where immediate access to information required, Serves as the basis for identifying database requirements during system design.

Uses of Data Dictionary:

1. To manage the details in large systems.

2. To communicate a common meaning for all system elements.

3. To Document the features of the system.

4. To facilitate analysis of the details in order to evaluate characteristics and determine where system changes should be made.

**7. SYSTEM TESTING**

System Testing is a process of executing a program with the explicit intention of finding errors, which cause program failure. There are two general strategies for testing software. They are :

**7.1Code Testing**

**7.2 Specification testing**

**7.1 CODE TESTING**

This strategy examines the logic of a program and has been carried out to identify three levels of correctness of programs. Possible correctness is first achieved by giving arbitrary inputs. Then the inputs are carefully selected to obtain predicted output. This gives the probable correctness. All potentially problematic areas are checked in this way for the software to achieve probable correctness. Absolute correctness can be demonstrated by a test involving every possible combination of inputs. However, this cannot be performed with the software but to the existence of the various possible combinations of the inputs and due to time restrictions.

**7.2 SPECIFICATION TESTING**

The specifications are examined which states what the program should do and how it should perform under various conditions. Then test cases are developed for each condition or combinations of conditions and submitted for processing. By examining the results, it is determined whether the program performs according to its specified requirements.

**7.3 LEVELS OF TESTING**

The two levels of Testing are

* Unit Testing
* System Testing

**7.3.1 UNIT TESTING**

Unit testing is done for the programs making up the systems. It is focused to find out module errors and enables to detect errors in coding and logic that are contained in the module. Unit testing is performed from bottom-up, starting with the smallest and lowest levels modules and proceeding one.

**7.3.2 SYSTEM TESTING**

At a time System Testing finds out the discrepancies between the system and its original objective, current specifications and systems documentation.The training session consists of getting the users used to software by asking them to perform data entry in our presence and look into the problems if encountered .

Testing can be done in two ways.

1. Sample Tests

2. Real Tests

**7.4 SAMPLE TESTS**

The software was tested with sample data that we randomly selected. I tested all functions with such random data and I was successful in getting accurate results. It was at this time I got to know certain intricacies of the system that I had overlooked.Without much delay however, I got over the problems and managed to perfect the software at least to the extent possible.

**7.5 REAL TEST**

For the real test, I have planned to do in due course. I initialized the software and creation of entities through the updation module, transaction entries through the transaction module and generated reports with the estimations. The various information retrieval functions as per user need are also implemented

**8. CONCLUSION AND FUTURE ENHANCEMENTS**

**8.1. CONCLUSION:**

**The main objective of the project is to access the robot from any region and to keep track of the robot and make it do required functions by just controlling it form home or workstation .The robot can record values like temperature ,humidity and many other constraints and even record the surrounding areas which helps in spying as well. The design is made in consideration that it can be used by any profession person without any confusion The efficiency of this system designed to suit an organization depends cooperation during the implementation stage and also flexibility of the system to adopt itself to the organization.**

**8.2 FUTURE ENHANCEMENTS:**

**The entire project has been developed and deployed as per requirements stated by the user, it is found to be bug free as per the testing standards that are implemented.. Any specification untraced errors will be concentrated in coming versions, which are planned to be developed in near future.**

**This system is ever ready to attend the following needs of future:**

**- Interaction with external software.**

**- Cross platform functionality**

**- Any other feature that client needs**

**9. REFERENCES**

* *Getting Started with Arduino*; Massimo Banzi, Michael Shiloh; 262 pages; 2014; [ISBN 1-4493-6333-4](https://en.wikipedia.org/wiki/Special:BookSources/1449363334).
* *Make: Sensors*; Tero Karvinen, Kimmo Karvinen, Ville Valtokari; 400 pages; 2014; [ISBN 978-1-4493-6810-4](https://en.wikipedia.org/wiki/Special:BookSources/9781449368104).
* *Arduino For Dummies*; John Nussey; 446 pages; 2013; [ISBN 978-1118446379](https://en.wikipedia.org/wiki/Special:BookSources/9781118446379).
* *Programming Arduino Next Steps: Going Further with Sketches*; Simon Monk; 2013; [ISBN 978-0071830256](https://en.wikipedia.org/wiki/Special:BookSources/9780071830256).
* *Exploring Arduino: Tools and Techniques for Engineering Wizardry*; Jeremy Blum; 384 pages; 2013; [ISBN 978-1118549360](https://en.wikipedia.org/wiki/Special:BookSources/9781118549360).
* *Arduino Workshop: A Hands-On Introduction with 65 Projects*; John Boxall; 392 pages; 2013; [ISBN 978-1593274481](https://en.wikipedia.org/wiki/Special:BookSources/9781593274481).
* *Beginning C for Arduino: Learn C Programming for the Arduino and Compatible Microcontrollers*; Jack Purdum; 280 pages; 2012; [ISBN 978-1430247760](https://en.wikipedia.org/wiki/Special:BookSources/9781430247760).
* *Programming Arduino: Getting Started With Sketches*; Monk Simon; 162 pages; 2011; [ISBN 978-0071784221](https://en.wikipedia.org/wiki/Special:BookSources/9780071784221).
* *Make: Electronics (Learning by Discovery)*; Charles Platt; 352 pages; 2009; [ISBN 978-0596153748](https://en.wikipedia.org/wiki/Special:BookSources/9780596153748).