# 1. INTRODUCTION

## 1.1. Introduction to Embedded Systems

Now a days every process is getting automated. This automation of the processes is possible with the help of embedded system. The process of connecting various devices and hardware together to make them act as a single system is embedded systems. The main advantage of the usage of embedded systems is the labor of the human in performing tasks get reduced.

There are various applications of embedded system in various fields like security, agriculture, home needs.

Watering plant is a daily activity which we do in our lives. There are two aspects which are to be considered while watering a plant. They are, firstly, we need to know at what time we must water a plant and secondly, what is the amount of water to be given to the plant.

Plant watering becomes difficult and challenging when the person goes on a trip or vacation. This leads to non-watering a plant and thereby dryness of plant. This can be overcome by the system which takes care of watering. That means this process must be automated.

Another problem of watering a plant manually is overwatering. During rainy or winter seasons, frequent watering may lead to overwatering and thereby death of plant. Again this problem needs a solution which is again automation of this process. So that the system takes care of the time at which the plant has to be watered and time for which a plant has to be watered.

With this project this is totally converted from manual process to an automated one in which we can reduce the disorders caused by watering a plant manually.

For this firstly, user needs to know the moisture levels in the soil to know the amount of time for which we need to water a plant. We use a sensor which senses the humidity and temperature of the atmosphere.

Secondly, the system needs to send the humidity and temperature details to the user through a mobile application. So that the user can fix the time for which the plant has to be watered. The user can do this using the same application.

The time thus fixed by the user must be received by the system and must be given to the motor in order to turn on the water into the plant. This entire system makes the job easier to the gardener and safe to the plant.

## 1.2. Existing system

An automatic drip irrigation system with an automatic timer which controls the frequency and run-time of water for the plants. This battery operated timer, once set, can take care of plant-watering automatically.

Considering the fact that the more and more towns/villages are moving towards urbanization, balcony gardening or roof-terrace gardening is going to be an integral part of our daily life.

At the same time, because of the busy schedules of metro life, people don't find time to water the plants regularly. Watering the plants becomes even more of a headache while going on vacations.

### *1.2.1. Disadvantages of Existing System*

- Over watering
- Fixed time period
- More frequency
- No change during suitable climate

## 1.3. Proposed system

To overcome over watering, we include the time management system which can be achieved by setting the time manually through the mobile. We can have a different number of time periods to water the plants, through which we can overcome the fixed frequency problem in the former system. The amount of water to be given to the plant is controlled by the mobile through a message using GSMmodule. By using sensors we can overcome the high frequency problem. By

using the sensors we can able to change the period up to which a plant has to be watered.

### *1.3.1. Advantages of Proposed System*

- Flexibility.
- No non-watering.
- No over watering.
- Manageable time period.
- Manageable frequency

# 2. SYSTEM ANALYSIS

Requirement analysis is a software engineering task that bridges the gap between system software allocation and software design. It provides system engineer to specify software function and performance indicate software's interface with other system elements and establish constraints that software must need. The basic is to obtain a clear picture of the needs and requirements of the end user and also the organization.

## 2.1. Hardware and Software requirements

### *Software Requirements*

In addition to the hardware requirements, certain software is also essential for the successful execution of the project. In fact they can be considered as the heart of the project. They vary from project to project. One set of software requirements exist for web-based projects whereas other set exist for embedded ones and so on.

For this project they include

- Arduino Software
- Programming Language : Embedded C++
- Xml
- Java

### *Hardware Requirements*

When you plan for the hardware that is required for a Project, as a starting point, you should determine the usage requirements for your Project environment. These variables include the number of projects, tasks, users, average tasks per project, and so on.

For this project they include

- Arduino : UNO
- GSM Module : 900
- Soil moisture sensor
- Relay Module
- Patch chords

## 2.2. Software Requirement Specification (SRS)

### *2.2.1. Vision*

For the people or gardener who would like to reduce their efforts in watering the plants manually, automated plant watering is an automated system that provides a facility of watering plants to help them grow healthier by avoiding non watering and over watering.

Unlike the existing system, our product not only waters the plants but also takes care of the amount of water to be given to the plants.

### *2.2.2. System Features and Functional Capabilities*

1. The system should able to water the plants automatically
2. The systems must be able to sense the moisture levels in the soil.
3. The system must be able to water the plant with appropriate amount of water.
4. The system must be able to reduce the efforts of human by\automation.

### *2.2.3. Functional Requirements*

1. The system should able to water the plants automatically.
2. The systems must be able to sense the moisture levels in the soil.
3. The system must be able to water the plant with appropriate amount of water.
4. The system must be able to reduce the efforts of human by automation.

*2.2.4. Non Functional Requirements*

1. The user must be able to understand the message sent by the arduino regarding humidity and temperature.
2. The user must be able to send the message from the mobile application to the GSM module.
3. The user must be able to operate the mobile application.

*2.2.5. Scope*

**2.2.5.1. Interfacing sensor to Arduino:**

- DHT11 senses the humidity and temperature.
- Using data pin it sends the values of humidity and temperature to digital pin of Arduino

**2.2.5.2. Interfacing GSM to Arduino:**

- Sends the humidity and temperature values from arduino through GSM to mobile app.
- Receives the message from mobile app through GSM as specified by the user in terms of time.

**2.2.5.3. Mobile application:**

- Imports the message from mobile SIM to mobile application.
- Type reply message on mobile app and send to GSM using mobile SIM number.

**2.2.5.4. Interfacing motor and Arduino:**

- Receives the message from arduino
- Switches on the motor

- Delays for specified time
- Switches off the motor

*2.2.6. Components*

**2.2.6.1. List of Actors**

- Gardener
- Arduino
- GSM
- Relay

**2.2.6.2. List of Use Cases**

**Gardener**

- Receives humidity and temperature values.
- Set the time for which the plant has to be watered.

**Arduino**

- Sends the message regarding the humidity and temperature in the atmosphere
- Receives the message from mobile in terms of time for which the plant has to be watered

**GSM**

- Sends the message regarding the time set by the user
- Receives the message from arduino regarding humidity and temperature in the atmosphere.

**Relay**

- Turns on the motor
- Turns off the motor

### 2.2.6.3. *Detailed description of use cases*

| Use case name | Identify humidity and temperature |
|---|---|
| **Use case id** | UC01 |
| **Actors** | Gardener |
| **Goal** | Notifies the humidity and temperature values in atmosphere. |
| **Summary** | Gardener receives the data through mobile application |
| **Preconditions** | The user must install the mobile application in the mobile for which the message is expected. |
| **Post conditions** | Successful data receiving. |

Table 2.1.Identifying humidity and temperature

| Use case name | Set time |
|---|---|
| **Use case id** | UC02 |
| **Actors** | Gardener |
| **Goal** | Allows the user to set the time for which the plant has to be watered. |
| **Summary** | The member gives the time in the space provided in the mobile application. |
| **Preconditions** | The user must have an idea regarding the general range of humidity and temperature to water a plant. |
| **Post conditions** | Successful setting of time. |

Table 2.2.Setting time

| Use case name | Send message to mobile using GSM |
|---|---|
| **Use case id** | UC03 |
| **Actors** | Arduino |
| **Goal** | Allows the arduino to send the message regarding humidity and temperature to mobile using GSM. |
| **Summary** | The member gives the information about humidity and temperature to the user through mobile app. |
| **Preconditions** | The arduino must receive the data from DHT11 before sending it to the mobile. |
| **Post conditions** | Successful transmission of data. |

Table 2.3.Send message to mobile using GSM

| Use case name | Read message received by GSM |
|---|---|
| **Use case id** | UC04 |
| **Actors** | Arduino |
| **Goal** | Allows the arduino to read the message received from GSM regarding time set by the gardener. |
| **Summary** | The member reads the data received from the user through GSM (time). |
| **Preconditions** | The arduino must receive the message from the mobile through GSM. |
| **Post conditions** | Successful receiving of time. |

Table 2.4.Read message received by GSM

| Use case name | Send message to mobile |
|---|---|
| **Use case id** | UC05 |
| **Actors** | GSM |
| **Goal** | Allows the arduino to send message regarding temperature and humidity using SIM. |
| **Summary** | The member sends the humidity and temperature to the mobile app. |
| **Preconditions** | The GSM must be provided with a SIM which is able to send and receive message. |
| **Post conditions** | Successful transmission of message. |

Table 2.5.Send message to mobile

| Use case name | Receiving message from mobile |
|---|---|
| **Use case id** | UC06 |
| **Actors** | GSM |
| **Goal** | Allows the arduino to receive message regarding time using SIM. |
| **Summary** | The member receives time from the mobile app. |
| **Preconditions** | The GSM must be provided with a SIM which is able to send and receive message. |
| **Post conditions** | Successful receiving of time. |

Table 2.6.Receive message from mobile

| Use case name | Turn on motor |
|---|---|
| **Use case id** | UC07 |
| **Actors** | Motor |
| **Goal** | Allows the relay to switch on the motor to water the plant. |
| **Summary** | The member switches on the motor. |
| **Preconditions** | The relay must receive the instruction from arduino to switch on the motor. |
| **Post conditions** | Successful initialization of watering. |

Table 2.7.Turn on motor

| Use case name | Turns off motor |
|---|---|
| **Use case id** | UC08 |
| **Actors** | Motor |
| **Goal** | Allows relay to switch off the motor after time out. |
| **Summary** | The member switches off the motor after time off. |
| **Preconditions** | The relay must receive the instruction from arduino to switch off the motor after time out. |
| **Post conditions** | Successful completion of watering. |

Table 2.8.Turn off motor

# 3. SYSTEM DESIGN

## 3.1. Introduction

System design is the process of defining the architecture, components, modules, interfaces, and data for a system to satisfy specified requirements. Systems design could be seen as the application of systems theory to product development. There is some overlap with the disciplines of systems analysis, systems architecture and systems engineering.

## 3.2. Data Dictionary

After carefully understanding the requirements of the candidate the entire data storage requirements are divided into tables. The below tables are normalized to avoid any anomalies during the course of data entry.

The volume of data in most information system application is substantially more than a single user can easily keep track of data dictionaries are an integral component of structural analysis, since data flow diagrams by themselves do not fully describe the subject of investigation .The data dictionary provides additional information about the system.

**What is Data Dictionary?**

A Data Dictionary a repository of the element in a system .As the name suggests, these elements center around data and the way they are structured to meet user requirements and organization needs. In a data dictionary we will find a list of all elements composing the data flowing through a system .The major elements are data flows, data stores and process the Data Dictionary stores details and descriptions of these elements.

**Why is a data dictionary important?**

Analysts use data dictionaries for the following reasons:

- To manage the details in large systems.
- To communicate a common meaning for all elements.
- To document the features of the system.
- To locate errors and omissions in the system.

## 3.3. UML Diagram

The Unified Modeling Language allows the software engineer to express an analysis model using the modeling notation that is governed by a set of syntactic semantic and pragmatic rules. A UML system is represented using five different views that describe the system from distinctly different perspective. Each view is defined by a set of diagram, which is as follows.

Use case diagram represents the functionality of the system from a user's point of view. Use cases focuses on the behavior of the system from external point of view. The unified modeling language allows the software engineers to express an analysis model using the modeling notation that is governed by a set of syntactic semantic and pragmatic rules

A UML system is represented using four different views that describe the system from distinctly perspective. Each view is defined by set diagrams, which is as follows.

**User Model View**

This view represents the system from the user's perspective. The analysis representation describes a usage scenario from the end-user perspective.

**Structural Model View**

In this model the data and functionality are arrived from inside the system. This model view models the static structures.

**Behavioral Model View**

It represents the dynamic of behavioral as parts of the system, depicting the interactions of collection between various structural elements described in the user model and structural model view.

**Implementation Model View**

In this the structural and behavioral as parts of the system are represented as they are to be built.

### *3.3.1. Use case diagram*

A use case diagram at its simplest is a representation of a user's interaction with the system that shows the relationship between the user and the different use cases in which the user is involved. A use case diagram can identify the different types of users of a system and the different use cases and will often be accompanied by other types of diagrams as well.

While a use case itself might drill into a lot of detail about every possibility, a use-case diagram can help provide a higher-level view of the system. It has been said before that "Use case diagrams are the blueprints for your system". They provide the simplified and graphical representation of what the system must actually do.

Due to their simplistic nature, use case diagrams can be a good communication tool for stakeholders. The drawings attempt to mimic the real world and provide a view for the stakeholder to understand how the system is going to be designed.

The purpose of the use case diagrams is simply to provide the high level view of the system and convey the requirements in layman's terms for the stakeholders. Additional diagrams and documentation can be used to provide a complete function.

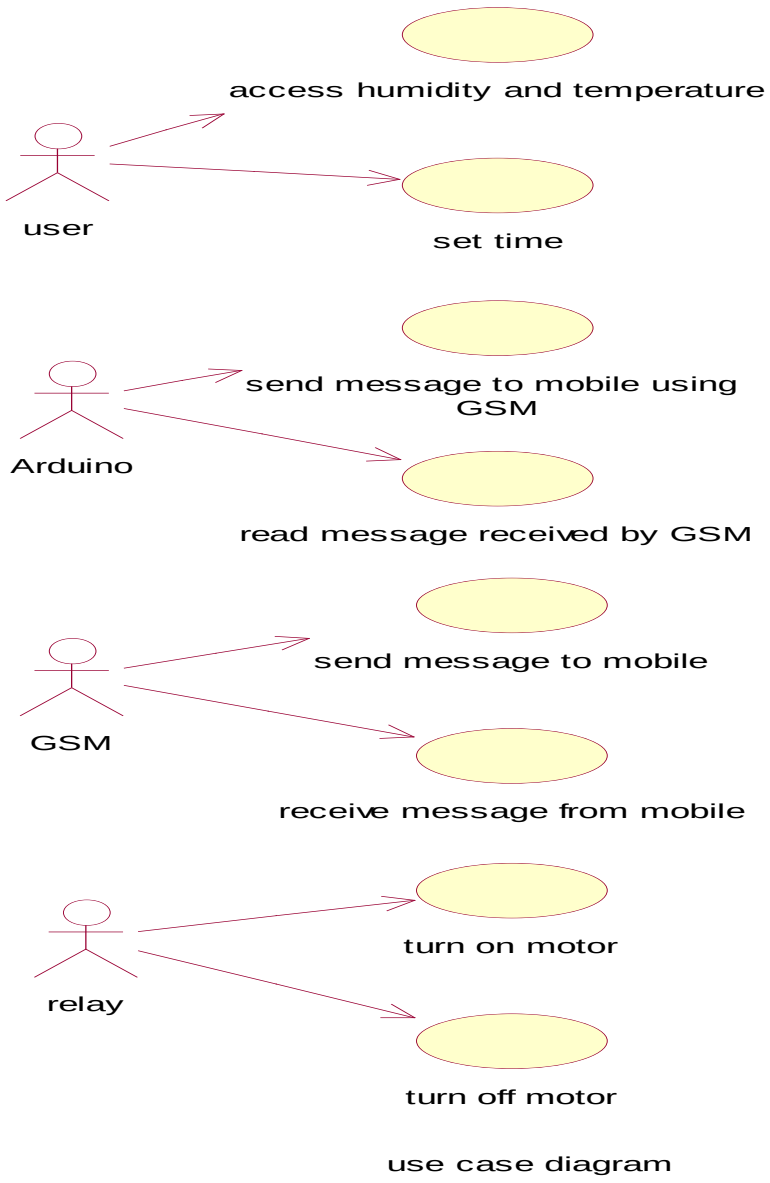



Figure 3.1. Use case diagram

### *3.3.2. Class diagram*

A class diagram is a picture for describing generic descriptions of possible systems. Class diagrams and Collaboration diagrams are alternate representations of object models. Class diagrams contain classes and object diagrams contain objects, but it is possible to mix classes and objects when dealing with various kinds of metadata, so the separation is not rigid.

Class diagrams are more prevalent than object diagrams. Normally you will build class diagrams plus occasional object diagrams illustrating complicated data structures or message-passing structures.

Class diagrams contain icons representing classes, interfaces, and their relationships. You can create one or more class diagrams to depict the classes at the top level of the current model; such class diagrams are themselves contained by the top level of the current model. You can also create one or more class diagrams to depict classes contained by each package in your model; such class diagrams are themselves contained by the package enclosing the classes they depict; the icons representing logical packages and classes in class diagrams.

A dependency is a semantic connection between dependent and independent model elements.[5] It exists between two elements if changes to the definition of one element (the server or target) may cause changes to the other (the client or source). This association is unidirectional.

An association represents a family of links. A binary association (with two ends) is normally represented as a line. An association can link any number of classes. An association with three links is called a ternary association. An association can be named, and the ends of an association can be adorned with role names, ownership indicators, multiplicity, visibility, and other properties.

A Class diagram gives an overview of a system by showing its classes and the relationships among them. UML class is a rectangle divided into: class name, attributes, and operations.

Our class diagram has three kinds of relationships:

- Association
- Aggregation
- Generalization

Dependency–properties of one class depend on properties of another class.

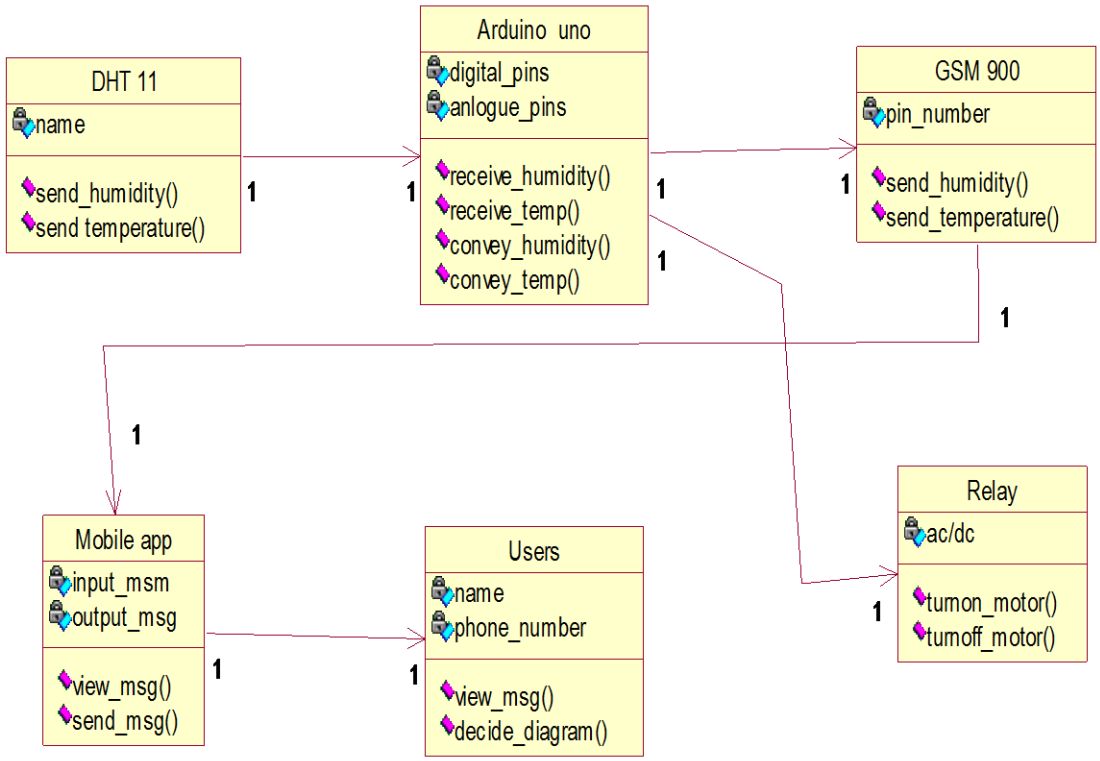



Figure 3.2. Class diagram

### *3.3.3. Sequence Diagram*

A sequence diagram is a graphical view of a scenario that shows object interaction in a time-based sequence, what happens next. Sequence diagrams establish the roles of objects and help provide essential information to determine class responsibilities and interfaces. This type of diagram is best used during early analysis phases in design because they are simple and easy to comprehend. Sequence diagrams are normally associated with use cases.

A Sequence diagram is an interaction diagram that shows how objects operate with one another and in what order. It is a construct of a message sequence chart.

A sequence diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. Sequence diagrams are typically associated with use case realizations in the Logical View of the system under development. Sequence diagrams are sometimes called event diagrams or event scenarios.

Sequence diagrams are closely related to collaboration diagrams and both are alternate representations of an interaction. There are two main differences between sequence and collaboration diagrams sequence diagrams show time-based object interaction while collaboration diagrams show how objects associate with each other.

Sequence diagram is an interaction diagram which focuses on the time ordering of messages. It shows a set of objects and messages exchange between these objects. This diagram illustrates the dynamic view of a system.

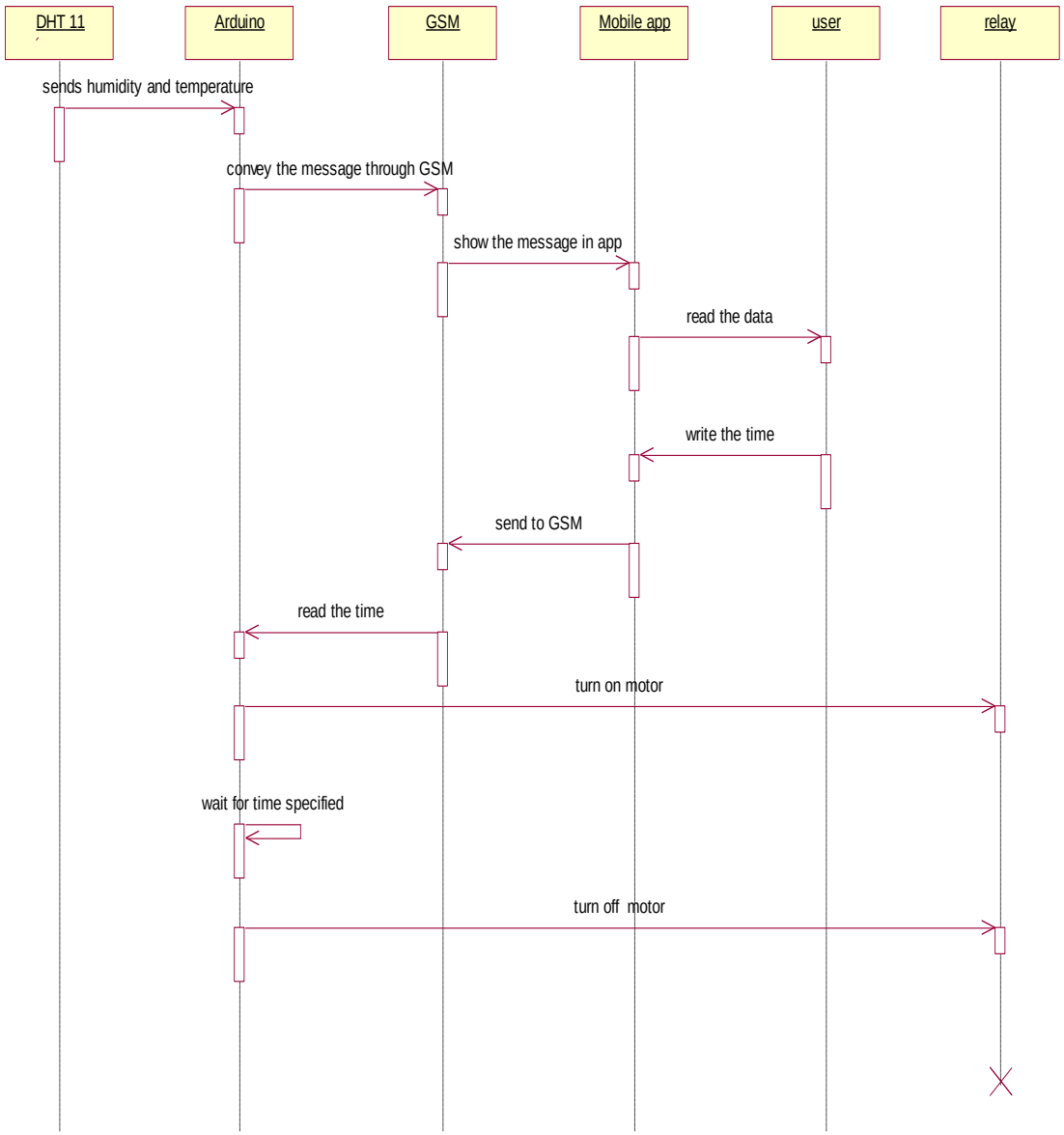



Figure 3.3. Sequence diagram

### *3.3.4. Collaboration Diagram*

Collaboration diagram is an interaction diagram which focuses on the structural organization of messages. It consists of objects which exchange information by using messages and has sequence numbers to the messages.

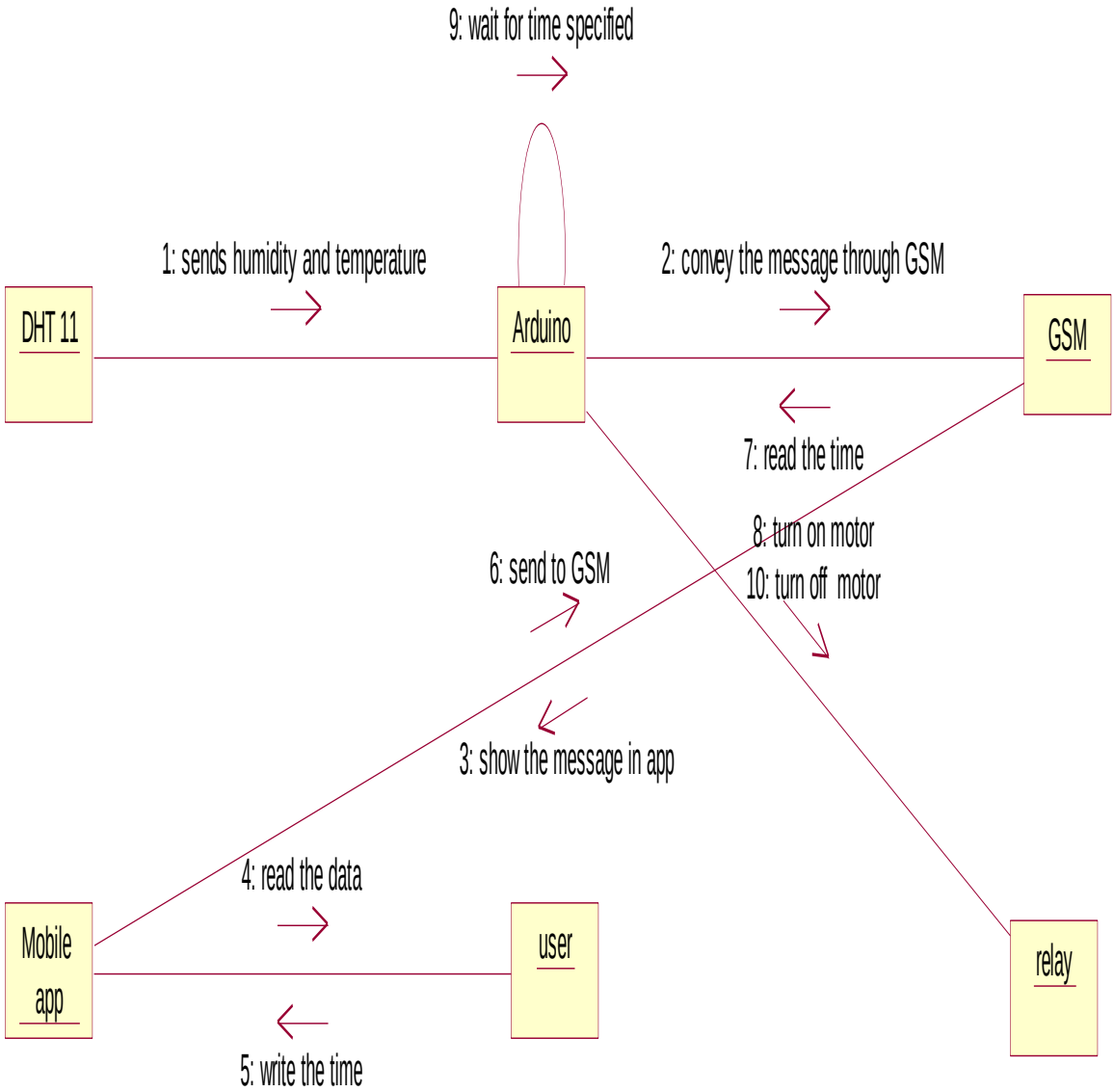



Figure 3.4. Collaboration diagram

# 4. SYSTEM IMPLEMENTATION

## 4.1. Introduction

In computer science, an implementation is a realization of a technical specification or algorithm as a program, software component, or other computer system through computer programming and deployment. Many implementations may exist for a given specification or standard.

## 4.2. Programming Languages and the Software of Hardware

### *4.2.1. Overview of embedded C++ and Arduino software*

Arduino is an open-source electronics platform based on easy-to-use hardware and software. Arduino boards are able to read inputs – light on a sensor, a finger on a button, or a Twitter message – and turn it into an output – activating a motor, turning on an LED, publishing something online. You can tell your board what to do by sending a set of instructions to the microcontroller on the board. To do so you use the Arduino programming language (based on Wiring), and the Arduino Software (IDE), based on Processing.

Over the years Arduino has been the brain of thousands of projects, from everyday objects to complex scientific instruments. A worldwide community of makers – students, hobbyists, artists, programmers, and professionals – has gathered around this open-source platform, their contributions have added up to an incredible amount of accessible knowledge that can be of great help to novices and experts alike.

***Features of arduino software***

A minimal Arduino C/C++ sketch, as seen by the Arduino IDE programmer, consist of only two functions:

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

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

## 4.3. Hardware Components

### *4.3.1.* Arduino

Arduino/Genuino Uno is a microcontroller board based on the Atmega328P (datasheet). It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz quartz crystal, a USB connection, a power jack, an ICSP header and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started.. You can tinker with your UNO without worrying too much about doing something wrong, worst case scenario you can replace the chip for a few dollars and start over again.


**Figure 4.1. Arduino UNO**

### *4.3.2.* GSM 900

The term GSM900 is used for a GSM system which operates in any 900 MHz The 900 MHz band defined in the ETSI standard includes the primary GSM band (GSM-P), the extension (see E-GSM) and the part of the 900 MHz band that is reserved for railways (R-GSM).

The total GSM900 band defined in the standard ranges from 876 – 915 MHz paired with 921 – 960 MHz Mobiles transmit in the lower band and base stations transmit in the upper band.

In daily life, the term GSM900 band is used for the parts of the band that are used by the GSM operators to offer public services, which excludes the R-

GSM band. This part of the band that remains ranges from 880 – 915 MHz paired with 925 – 960 MHz band.

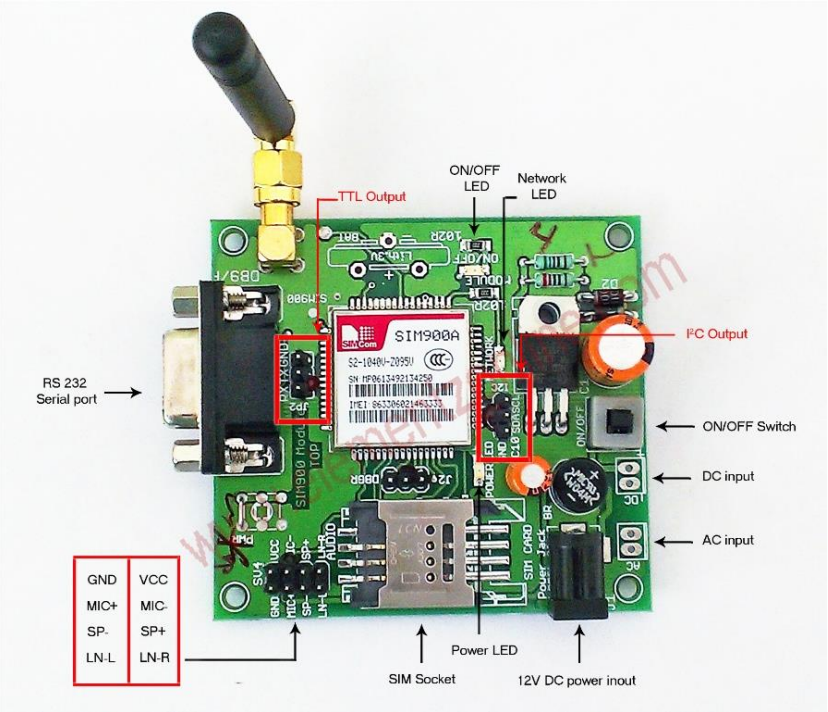



**Figure 4.2. GSM 900**

***Interfacing arduino and GSM900***

1. Insert the SIM card to module and lock it.

2. Connect the adapter to module and turn it ON!

3. Now wait for some time (say 1 minute) and see the blinking rate of 'status LED' (GSM module will take some time to establish connection with mobile network)

4. Once the connection is established successfully, the status LED will blink continuously every 3 seconds.

There are two ways of connecting GSM module to arduino. In any case, the communication between Arduino and GSM module is serial. So we are

supposed to use serial pins of Arduino (Rx and Tx). So if you are going with this method, you may connect the Tx pin of GSM module to Rx pin of Arduino and Rx pin of GSM module to Tx pin of Arduino.

Now connect the ground pin of arduino to ground pin of GSM module! So that's all! You made 3 connections and the wiring is over! Now you can load different programs to communicate with GSM module and make it work.

The problem with this connection is while programming. Arduino uses serial ports to load program from the Arduino IDE. If these pins are used in wiring, the program will not be loaded successfully to Arduino. So you have to disconnect wiring in Rx and Tx each time you burn the program. Once the program is loaded successfully, you can reconnect these pins and have the system working! To avoid this difficulty, I am using an alternate method in which two digital pins of arduino are used for serial communication. We need to select two PWM enabled pins of arduino for this method. So I choose pins 9 and 10 (which are PWM enabled pins). This method is made possible with the Software Serial Library of Arduino. Software Serial is a library of Arduino which enables serial data communication through other digital pins of Arduino. The library replicates hardware functions and handles the task of serial communication.

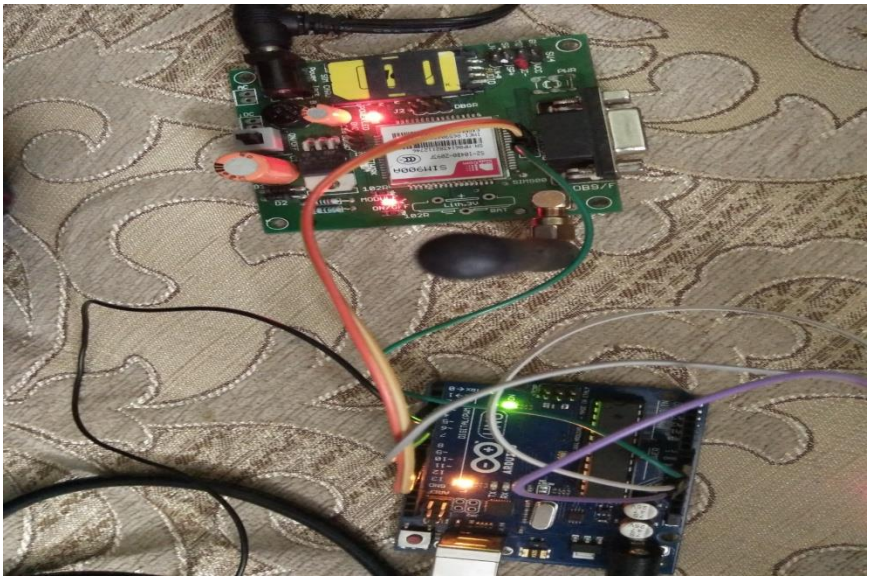


**Figure 4.3. Interfacing Arduino and GSM**

### *4.3.3.* DHT 11 sensor

The DHT11 is a basic, ultra-low-cost digital temperature and humidity sensor. It uses a capacitive humidity sensor and a thermistor to measure the surrounding air, and spits out a digital signal on the data pin (no analog input pins needed). It is fairly simple to use, but requires careful timing to grab data. The only real downside of this sensor is you can only get new data from it once every 2 seconds, so when using our library, sensor readings can be up to 2 seconds old.

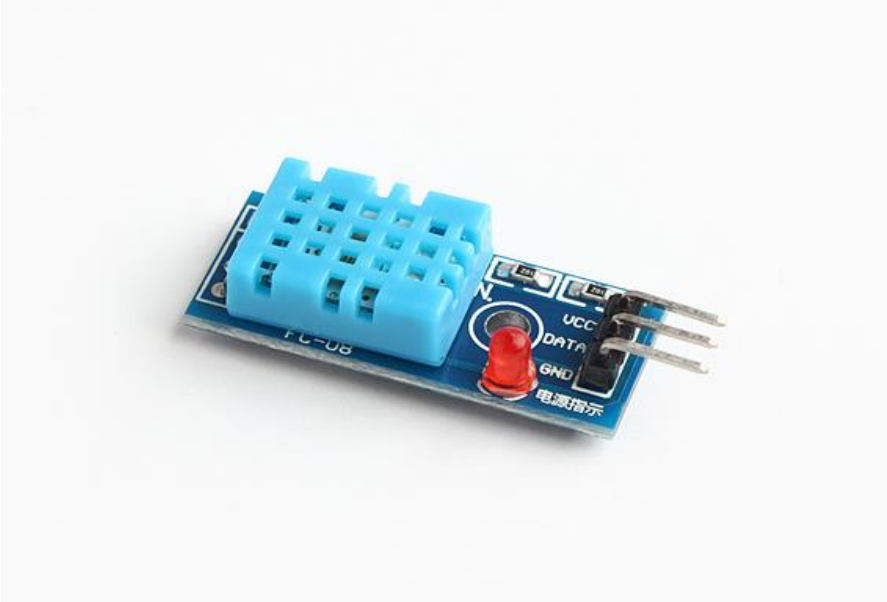



**Figure 4.4. DHT11 sensor**

***Interfacing arduino and DHT11 sensor:***

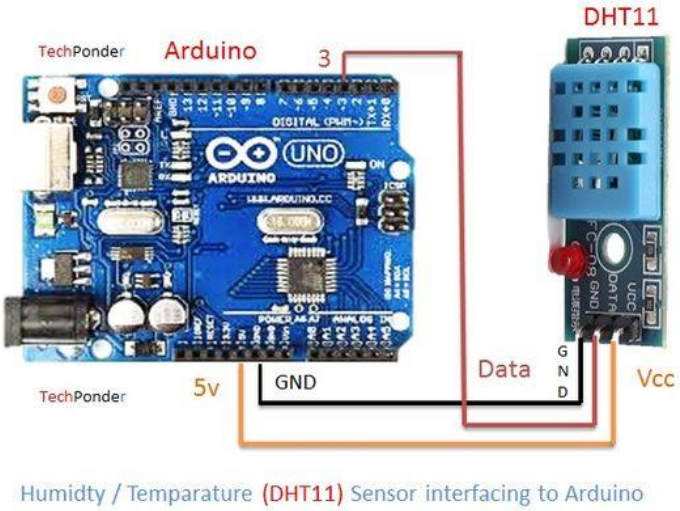



**Figure 4.5. Interfacing Arduino and DHT11 sensor**

### *4.3.4.* Relay


**Figure 4.6. Relay**

## 4.4. Software

### *4.4.1.* Android:

Android is a software stack for mobile devices that includes an operating system, middle ware and applications. The android SDK provides the tools and APIs necessary to begin developing applications on the android platform using java programming language.

Android is a Linux-based operating system designed primarily for touch screen mobile devices such as smart phones and tablet computers. Initially developed by Android, Inc., which Google backed financially and later bought in 2005, Android was unveiled in 2007 along with the founding of the Open Handset Alliance: a consortium of hardware, software, and telecommunication companies devoted to advancing open standards for mobile devices. The first Android-powered phone was sold in October 2008.

Android, Inc. was founded in Palo Alto, California in October 2003 by

•Andy Rubin (co-founder of Danger),

•Rich Miner (co-founder of Wildfire Communications, Inc.),

•Nick Sears (once VP at T-Mobile) and

•Chris White (headed design and interface development at WebTV).

### *4.4.2.* Features & Specifications

Android is a powerful Operating System supporting a large number of applications in Smart Phones. These applications make life more comfortable and advanced for the users. Hardware's that support Android are mainly based on ARM architecture platform. Some of the current features and specifications of android are:

Android comes with an Android market which is an online software store. It was developed by Google. It allows Android users to select, and download applications developed by third party developers and use them. There are around 2.0 lack+ games, application and widgets available on the market for users.

Android applications are written in java programming language. Android is available as open source for developers to develop applications which can be further used for selling in android market. There are around 200000 applications developed for android with over 3 billion+ downloads. Android relies on Linux version 2.6 for core system services such as security, memory management, process management, network stack, and driver model. For software development, Android provides Android SDK (Software development kit).

### *4.4.3.* Platform Architecture:

Android is an open source, Linux-based software stack created for a wide array of devices and form factors. The following diagram shows the major components of the Android platform

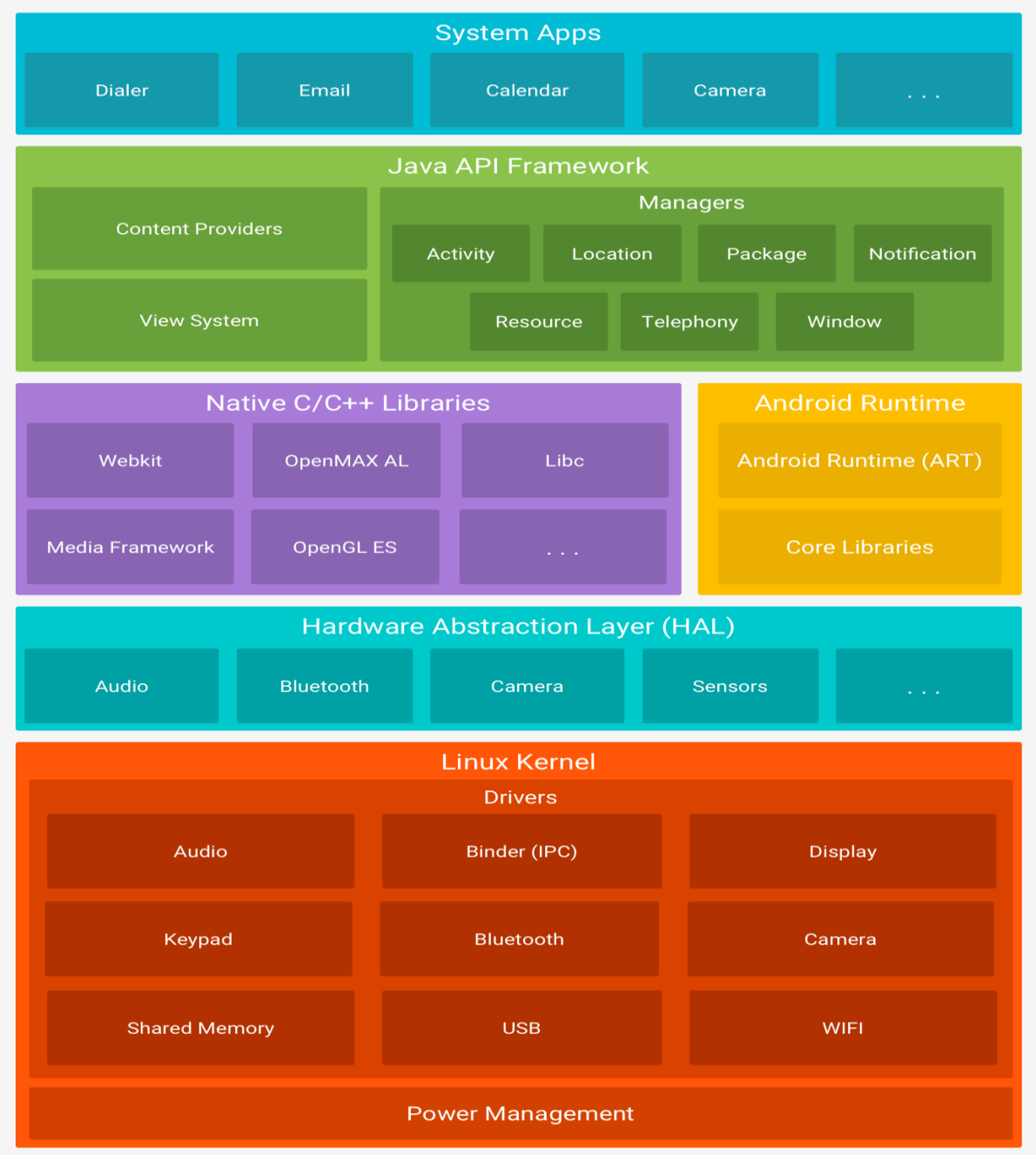



**Figure 4.7. Architecture of Android**

### *4.4.4.* The Linux Kernel

The foundation of the Android platform is the Linux kernel. For example, the Android Runtime (ART) relies on the Linux kernel for underlying functionalities such as threading and low-level memory management. Using a Linux kernel allows Android to take advantage of key security features and allows device manufacturers to develop hardware drivers for a well-known kernel.

### *4.4.5.* Hardware Abstraction Layer (HAL)

The hardware abstraction layer (HAL) provides standard interfaces that expose device hardware capabilities to the higher-level Java API framework. The HAL consists of multiple library modules, each of which implements an interface for a specific type of hardware component, such as the camera or Bluetooth module. When a framework API makes a call to access device hardware, the Android system loads the library module for that hardware component.

### *4.4.6.* Android Runtime

For devices running Android version 5.0 (API level 21) or higher, each app runs in its own process and with its own instance of the Android Runtime (ART). ART is written to run multiple virtual machines on low-memory devices by executing DEX files, a byte code format designed especially for Android that's optimized for minimal memory footprint. Build tool chains, such as Jack, compile Java sources into DEX byte code, which can run on the Android platform.

Some of the major features of ART include the following:

- Ahead-of-time (AOT) and just-in-time (JIT) compilation
- Optimized garbage collection (GC)
- Better debugging support, including a dedicated sampling profiler, detailed diagnostic exceptions and crash reporting, and the ability to set watch points to monitor specific fields

- Prior to Android version 5.0 (API level 21), Dalvik was the Android runtime. If your app runs well on ART, then it should work on Dalvik as well, but the reverse may not be true.
- Android also includes a set of core runtime libraries that provide most of the functionality of the Java programming language, including some Java 8 language features that the Java API framework uses.

### *4.4.7.* Native C/C++ Libraries

Many core Android system components and services, such as ART and HAL, are built from native code that require native libraries written in C and C++. The Android platform provides Java framework APIs to expose the functionality of some of these native libraries to apps. For example, you can access OpenGL ES through the Android framework's Java OpenGL API to add support for drawing and manipulating 2D and 3D graphics in your app. If you are developing an app that requires C or C++ code, you can use the Android NDK to access some of these native platform libraries directly from your native code.

### *4.4.8.* Java API Framework

The entire feature-set of the Android OS is available to you through APIs written in the Java language. These APIs form the building blocks you need to create Android apps by simplifying the reuse of core, modular system components and services, which include the following:

- A rich and extensible View System you can use to build an app's UI, including lists, grids, text boxes, buttons, and even an embeddable web browser
- A Resource Manager, providing access to non-code resources such as localized strings, graphics, and layout files
- A Notification Manager that enables all apps to display custom alerts in the status bar
- An Activity Manager that manages the lifecycle of apps and provides a common navigation back stack

- Content Providers that enable apps to access data from other apps, such as the Contacts app, or to share their own data

➢ Developers have full access to the same framework APIs that Android system apps use

### *4.4.9.* System Apps

Android comes with a set of core apps for email, SMS messaging, calendars, internet browsing, contacts, and more. Apps included with the platform have no special status among the apps the user chooses to install. So a third-party app can become the user's default web browser, SMS messenger, or even the default keyboard (some exceptions apply, such as the system's Settings app).The system apps function both as apps for users and to provide key capabilities that developers can access from their own app. For example, if your app would like to deliver an SMS message, you don't need to build that functionality yourself and you can instead invoke whichever SMS app is already installed to deliver a message to the recipient you specify.

### *4.4.10.* Benefits of Android

• Applications

- Google applications

Android includes most of the time many Google applications like Gmail, YouTube or Maps. These applications are delivered with the machine most of the time, except in certain cases, such as some Phones running android on which the provider has replaced Google applications by its own applications.

-widgets

With android, it is possible to use widgets which are small tools that can most often get Information. These widgets are directly visible on the main window.

-Android Market

This is an online software store to buy applications. Developers who created applications can add them into the store, and these applications can be downloaded by users, they can be both free and paid.

• Multitasking

Android allows multitasking in the sense that multiple applications can run simultaneously. With Task Manager it is possible view all running tasks and to switch from one to another easily.

• SDK

A development kit has been put at disposal of everybody. Accordingly, any developer can create their own applications, or change the android platform. This kit contains a set of libraries, powerful tools for debugging and development, a phone emulator, thorough documentation, FAQs and tutorials.

• Modifiability:

This allows everyone to use, improve or transform the functions of Android for example transform the interface in function of uses, to transform the platform in a real system embedded Linux. Following are the some of the benefits of android.

- Android applications are composed of one or more application components (activities, services, content providers, and broadcast receivers)
- Each component performs a different role in the overall application behavior, and each one can be activated individually (even by other applications)
- The manifest file must declare all components in the application and should also declare all application requirements, such as the minimum version of Android required and any hardware configurations required.
- Non-code application resources (images, strings, layout files, etc.) should include alternatives for different device configurations (such as different strings for different languages).

### *4.4.11.* Android Development Tools:

- **Android SDK:**

  The Android Software Development Kit (Android SDK) contains the necessary tools to create, compile and package Android applications. Most of these tools are command line based. The primary way to develop Android applications is based on the Java programming language.

- **Android Debug Bridge(ADB):**

  The Android SDK contains the Android Debug Bridge (ADB), which is a tool that allows you to connect to a virtual or real android device, for the purpose of managing the device or debugging your application.

- **Android Developer Tools and Android Studio:**

  Google provides two integrated development environments (IDEs) to develop new applications. The Android Developer Tools (ADT) are based on the Eclipse IDE. ADT is a set of components (plugins), which extend the Eclipse IDE with Android development capabilities. Google also supports an IDE called Android Studio for creating Android applications. This IDE is based on the Intelli JIDE. Both IDEs contain all required functionality to create, compile, debug and deploy Android applications. They also allow the developer to create and start virtual Android devices for testing. Both tools provide specialized editors for Android specific files.

  Most of Android's configuration files are based on XML. In this case these editors allow you to switch between the XML representation of the file and a structured user interface for entering the data. Dalvik Virtual Machine. The Android system uses a special virtual machine, i.e., the Dalvik Virtual Machine (Dalvik) to run Java based applications. Dalvik uses a custom bytecode format which is different from Java byte code. Therefore you cannot run Java class files on Android directly; they need to be converted into the Dalvik byte code format**.**

### *4.4.12.* The basics of creating applications:

To begin to program for Android we needed some basics, because some elements are very different, even if programming an application in Android uses the Java language, therefore, an object oriented language. Firstly, in an Android application, there is no main method. Following is the syntax of main method.

**Public static void main (String [] args) {...}**

This method that allows launching a program in java is not present in an application android.

## Activity:

An activity is a user interface that allows the user to interact with the screen, to perform actions. For example, a text messaging application could have an activity that displays a list of contacts to send messages. Once the contact is selected, activity could send information to a second activity that could serve to send the message to the contact. When an application is launched, what it displays is the result of an activity. At the code level, for create an activity, you must create a class that extends the Activity class. An activity has a required on Create () method. It is the main method. To interact with the program, through the activity, there must be something displayed, that is why the activity, contains what is called views.

## Intent:

An activity can of course start another one, even if it but to do this, it will need a special object called Intent. An intent is basis description of an operation to be performed. It can launch an Activity, send a broadcast Intent to any interested Broadcast Receiver components, and communicate with a background Service. An Intent performs binding between the codes in different applications. It can be thought of as the Link between activities. It is possible to add some information to an Intent. Following is the syntax to create an intent.

**Intent smsIntent = new Intent(Intent.ACTION_VIEW);**

## Android Manifest:

AndroidManifest.xml file is necessary for all android applications and must have this name in its root directory. In the manifest you can find essential information about the application for the Android system, information that the system must have before it can run any of the application's code. Here is what you can find in the Android manifest:

- The name of the Java package for the application. The package name serves as a unique identifier for the application.
- The description of the components of the application the activities, services, broadcast receivers and content providers that the application is composed of and under what conditions they can be launched.
- The processes that will host application components.
- The permissions the application must have in order to access protected parts of the API and interact with other applications.
- The permissions that others are required to have in order to interact with the application's16components.
- The list of the Instrumentation classes that provide profiling and other information as the application is running. These declarations are present in the manifest only while the application is being developed and tested; they're removed before the application is published.
- The minimum level of the Android API that the application requires.
- The list of the libraries that the application must be linked against.

### *4.4.13.* User Interface Representation

1. **Edit Text:**

Edit Text is a thin veneer over Text View that configures itself to be editable.

2. **Button:**

A button consists of text or an icon (or both text and an icon) that communicates what action occurs when the user touches it.

3. **List View:**

   List View is a view group that displays a list of scrollable items. The list items are automatically inserted to the list using an Adapter that pulls content from a source such as an array or database query and converts each item result into a view that's placed into the list.

4. **Card View:**

   A Frame Layout with a rounded corner background and shadow. Card View uses elevation property on Lollipop for shadows and falls back to a custom emulated shadow implementation on older platforms.

5. **Splash Screen:**

   An android application takes some time to start up, especially when the application is first launched on a device .A splash Screen may display startup progress to the user to indicate branding.

6. **Progress Bar:**

   Progress Bar is used to show progress of a task. In android there is a class called Progress Dialogue that allows you to create progress bar. In order to do this, you need to instantiate an object of this class.

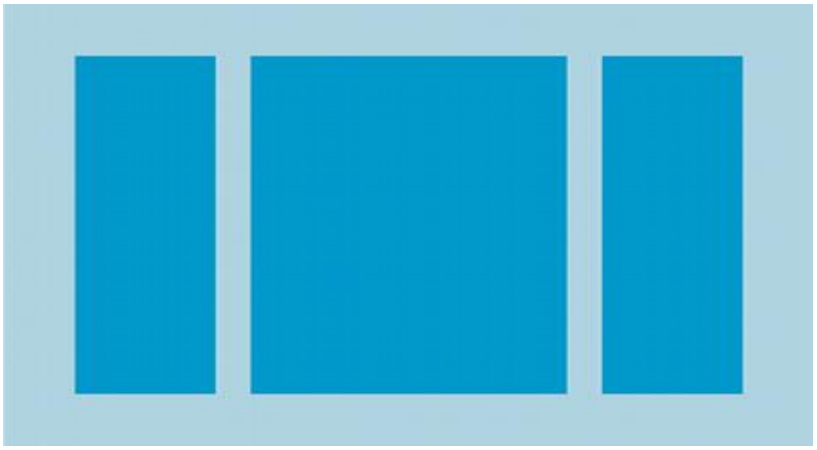
7. **Scroll View:**

   A scroll view is a Frame Layout container for a view hierarchy that can be scrolled by the user, allowing it to be larger than the physical display.

8. Linear Layout:

Linear Layout is a view group that aligns all the children in single direction, vertically or horizontally. You can specify the layout direction with the android: orientation attribute.

Following is the representation of Linear Layout.



9. **Relative Layout:**

Relative Layout is a view group that displays child views in relative positions. The position of each view can be specified as relative to sibling elements (such as to the left-of or below another view) or in positions relative to the parent relative layout area (such as aligned to the bottom, left or center).

Following is the representation of Relative Layout

### *4.4.14.* Overview of XML and java:

Xml means Extensible Markup Language. Android provides a straightforward XML vocabulary that corresponds to the View classes and subclasses. The goal of using Android's XML vocabulary, is to quickly design UI layouts and the screen elements they contain, in the same way that creating web pages in HTML with a series of nested elements.

Here is an example:

```
<? Xml version="1.0" encoding="utf-8"?>
        <LinearLayoutxmlns:android="http://schemas.android.com/apk/res/android"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:orientation="vertical" >
        <TextViewandroid:id="@+id/text"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello, I am a TextView" />
        <Button android:id="@+id/button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello, I am a Button" />
        </LinearLayout>
```

### Use of XML:

A View is the basic building block for user interface components. A View occupies a rectangular area on the screen. View is the base class for widgets, which are used to create interactive UI components (buttons, text fields, etc.).There are different kinds of views, for example a List View is able to display only an interactive list of what you want to display, while a Web View allows you to display a web page. As said before, a view occupies a rectangular area on the screen. To organize these rectangles on the screen, there is a text file written in XML for every different screen.

## Java:

Java is a very popular programming language developed by Sun Microsystems (now owned by Oracle).Developed long after C and C++, Java incorporates many of the powerful features of those powerful languages while addressing some of their drawbacks. Still, programming languages are only as powerful as their libraries. These libraries exist to help developers build applications.

Some of the Javas important core features are:

- It is easy to learn and easy.
- It is designed to be platform-independent and secure, using virtual machines.
- It is object-oriented.

## Platform Independence Important:

With many programming languages, you need to use a compiler to reduce your code down into machine language that the device can understand. While this is well and good, different devices use different machine languages. This means that you might need to compile your applications for each different device or machine language in other words, your code isn't very portable. This is not the case with Java. The Java compilers convert your code from human readable Java source files to something called byte code in the42Java world. These are interpreted by a Java Virtual Machine, which operates much like a physical CPU might operate on machine code, to actually execute the compiled code. Although it might seem like this is inefficient, much effort

has been put into making this process very fast and efficient. These efforts have paid off in that Java performance in generally second only to C/C++ in common language performance comparisons.

Android applications run in a special virtual machine called the Dalvik VM. While the details of this VM are unimportant to the average developer, it can be helpful to think of the Dalvik VM as a bubble in which your Android application runs, allowing you to not have to worry about whether the device is a Motorola Droid, an HTC Evo, or the latest toaster running Android. You don't care so long as the device is Dalvik VM friendly and that's the device manufacturer's job to implement, not yours.

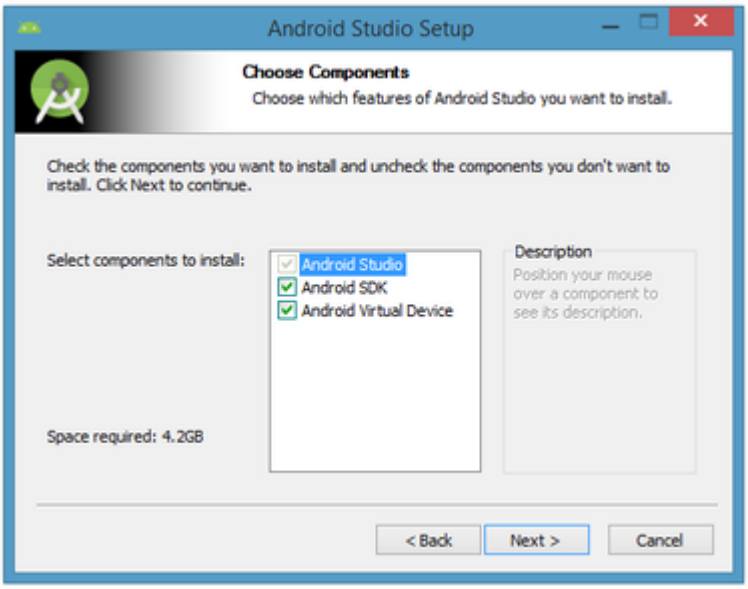
## Why is java secure?

Let's take this bubble idea a bit further. Because Java applications run within the bubble that is a virtual machine, they are isolated from the underlying device hardware. Therefore, a virtual machine can encapsulate, contain, and manage code execution in a safe manner compared to languages that operate in machine code directly. The Android platform takes things a step further. Each Android application runs on the (Linux- based) operating system using a different user account and in its own instance of the Dalvik VM. Android applications are closely monitored by the operating system and shut down if they don't play nice (e.g. use too much processing power, become unresponsive, waste resources, etc.).Therefore, it's important to develop applications that are stable and responsive. Applications can communicate with one another using well- defined protocol.

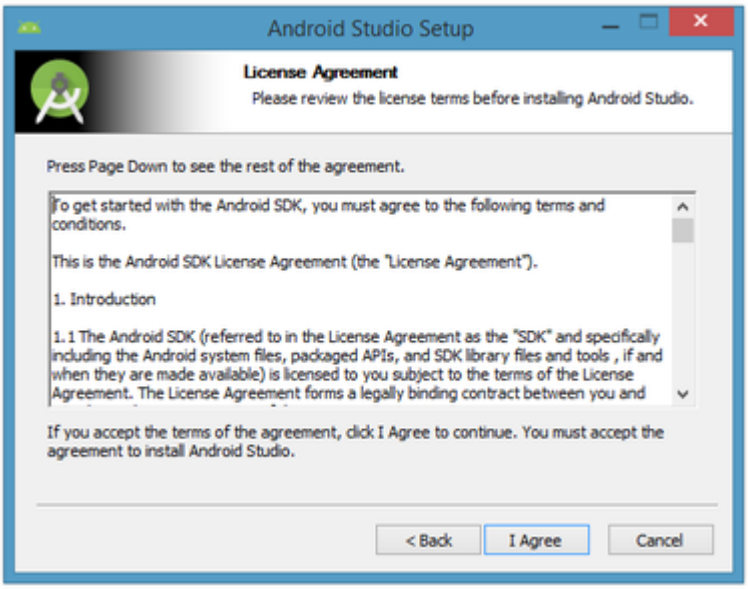
## Installation of Android studio:

I launched android-studio-bundle-143.2821654-windows.exe to start the installation process. The installer responded by presenting the Android Studio Setup dialog box shown in Figure 1.

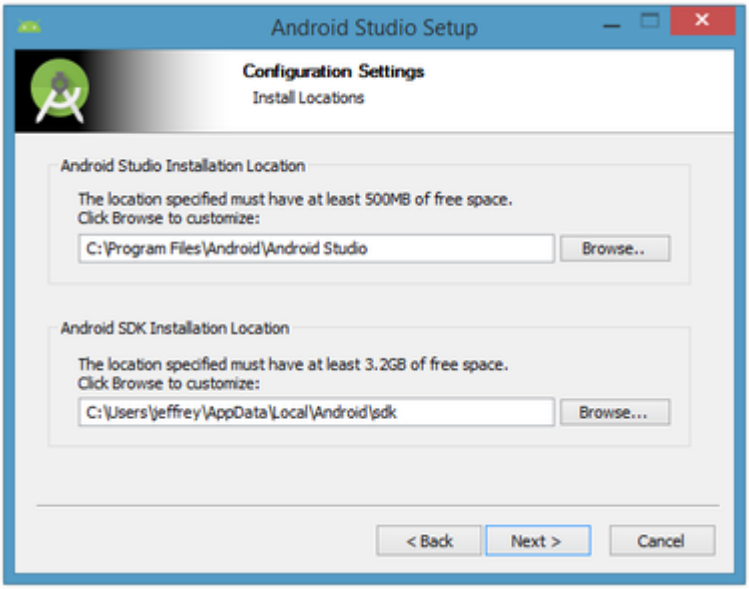

Clicking Next took me to the following dialog box, which gives you the option to decline installing the Android SDK (included with the installer) and an Android Virtual Device (AVD).



I chose to keep the default settings. After clicking next, you'll be taken to the license agreement dialog box. Accept the license to continue the installation.

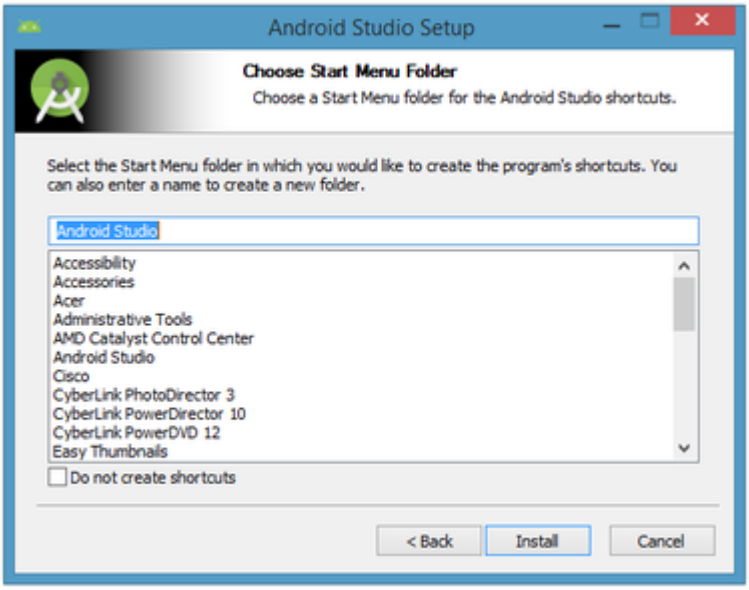



The next dialog box invites you to change the installation locations for Android Studio and the Android SDK.



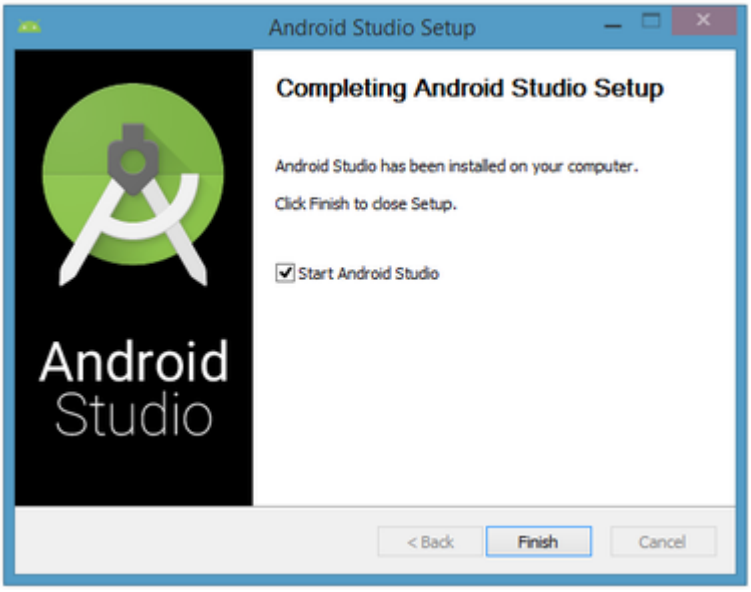

Change the location or accept the default locations and click next.

The installer defaults to creating a shortcut for launching this program, or you can choose to decline. I recommend that you create the shortcut, then click the Install button to begin installation.



The resulting dialog box shows the progress of installing Android Studio and the Android SDK. Clicking the Show Details button will let you view detailed information about the installation progress.

The dialog box will inform you when installation has finished. When you click next, you should see the following:



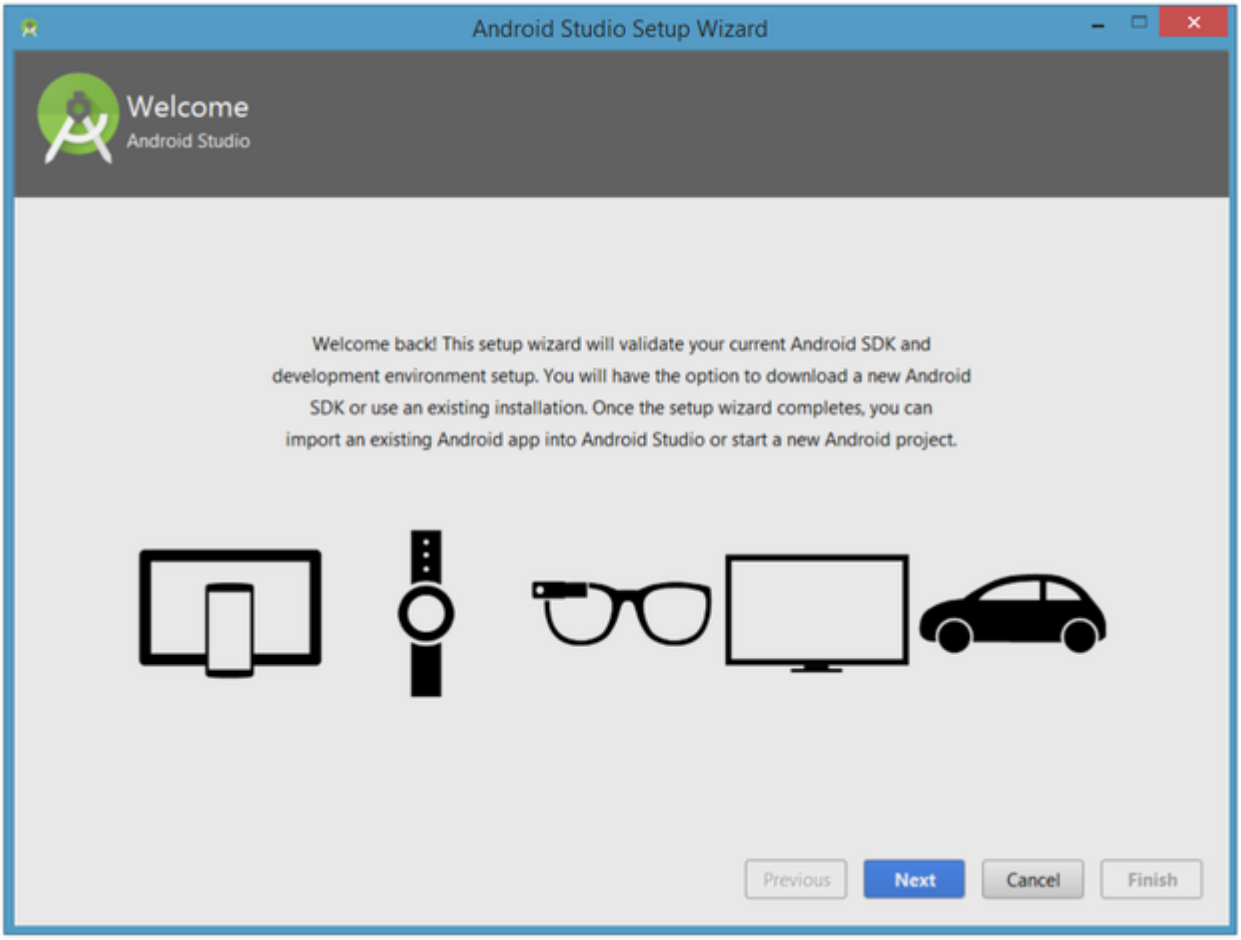

To complete your installation, leave the Start Android Studio box checked and click Finish.

## Running Android Studio

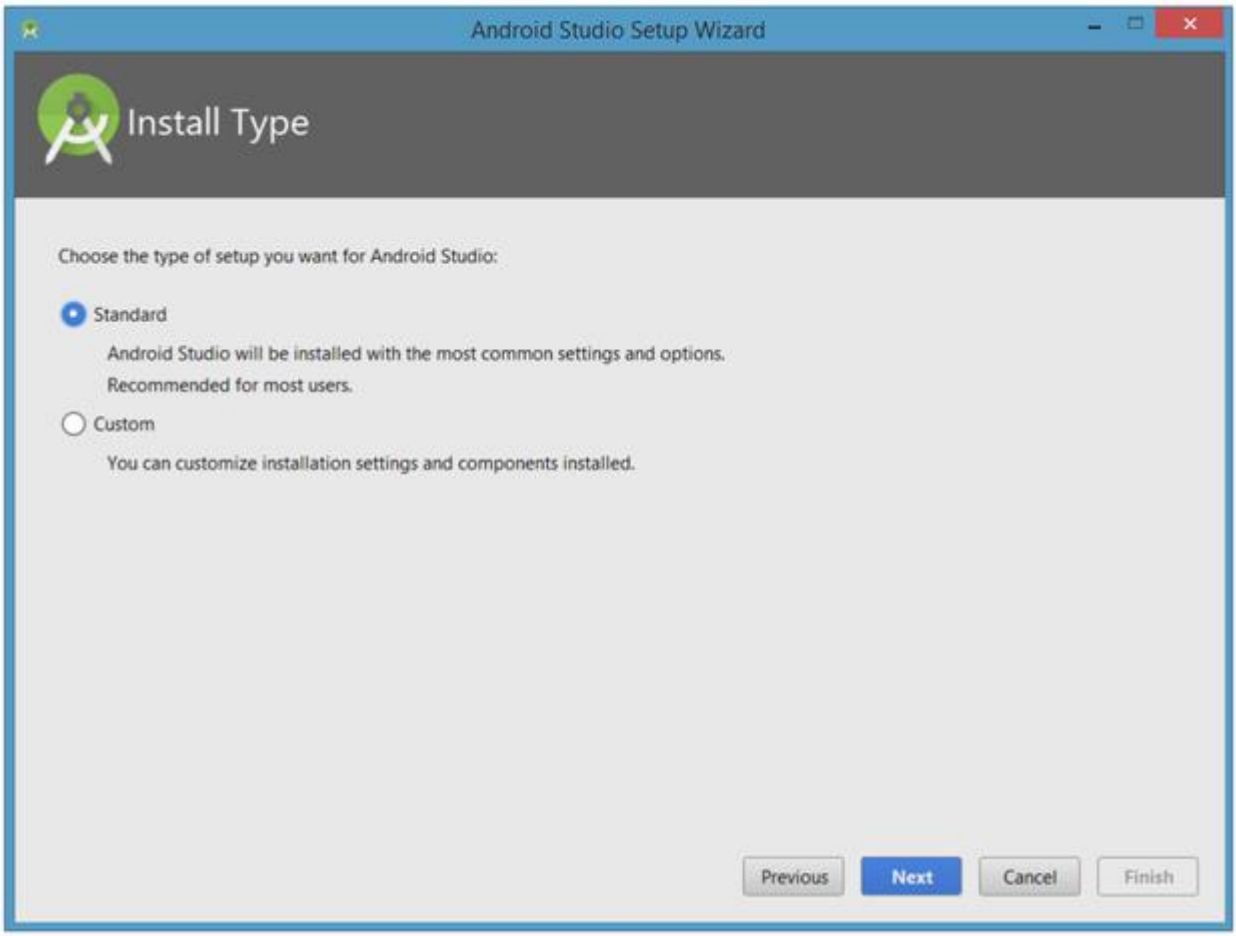
Android Studio presents a splash screen when it starts running:


On your first run, you'll be asked to respond to several configuration-oriented dialog boxes. The first dialog box focuses on importing settings from any previously installed version of Android Studio.

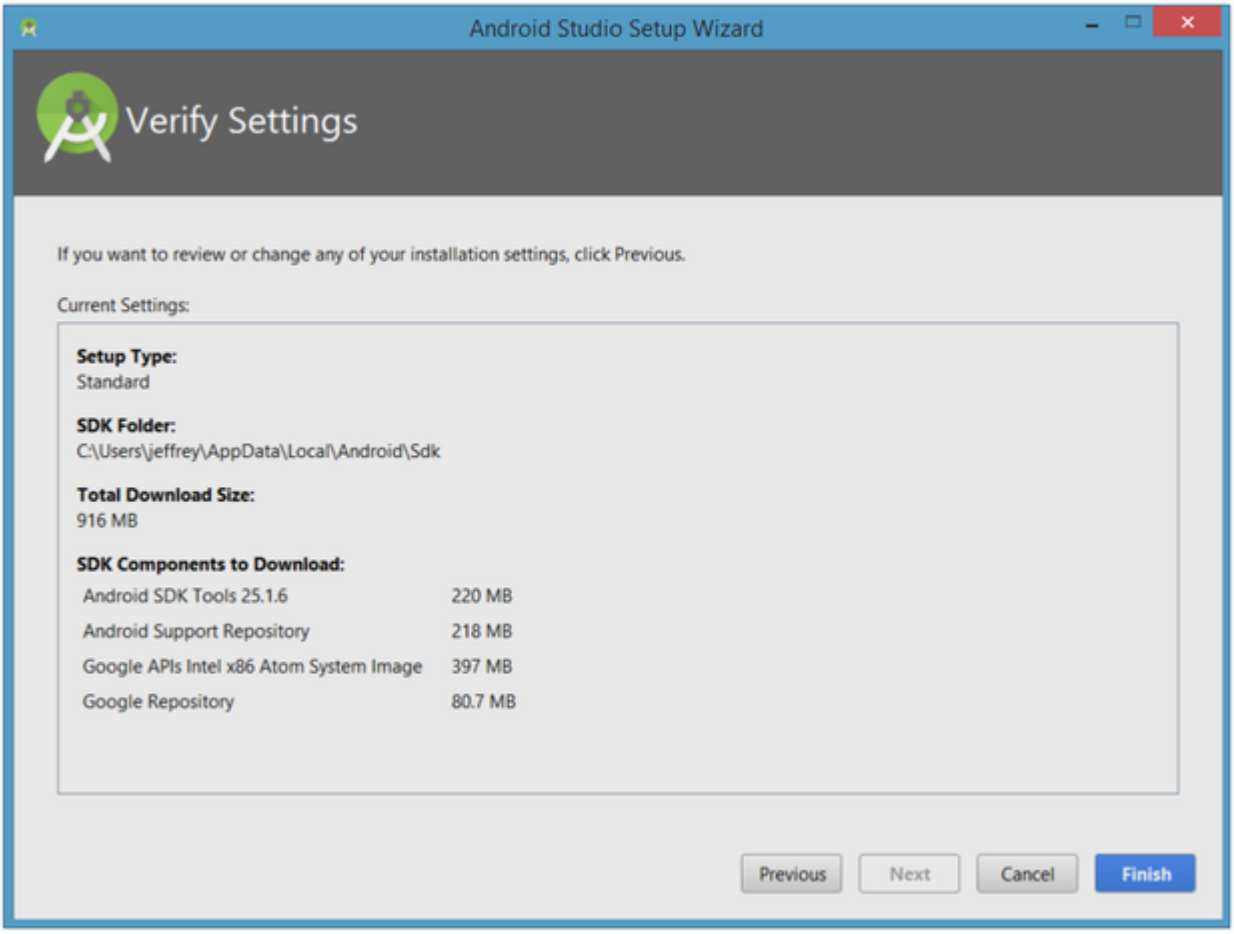

If you're like me, and don't have a previously installed version, you can just keep the default setting and click OK. Android Studio will respond with a slightly enhanced version of the splash screen, followed by the Android Studio Setup Wizard dialog box:



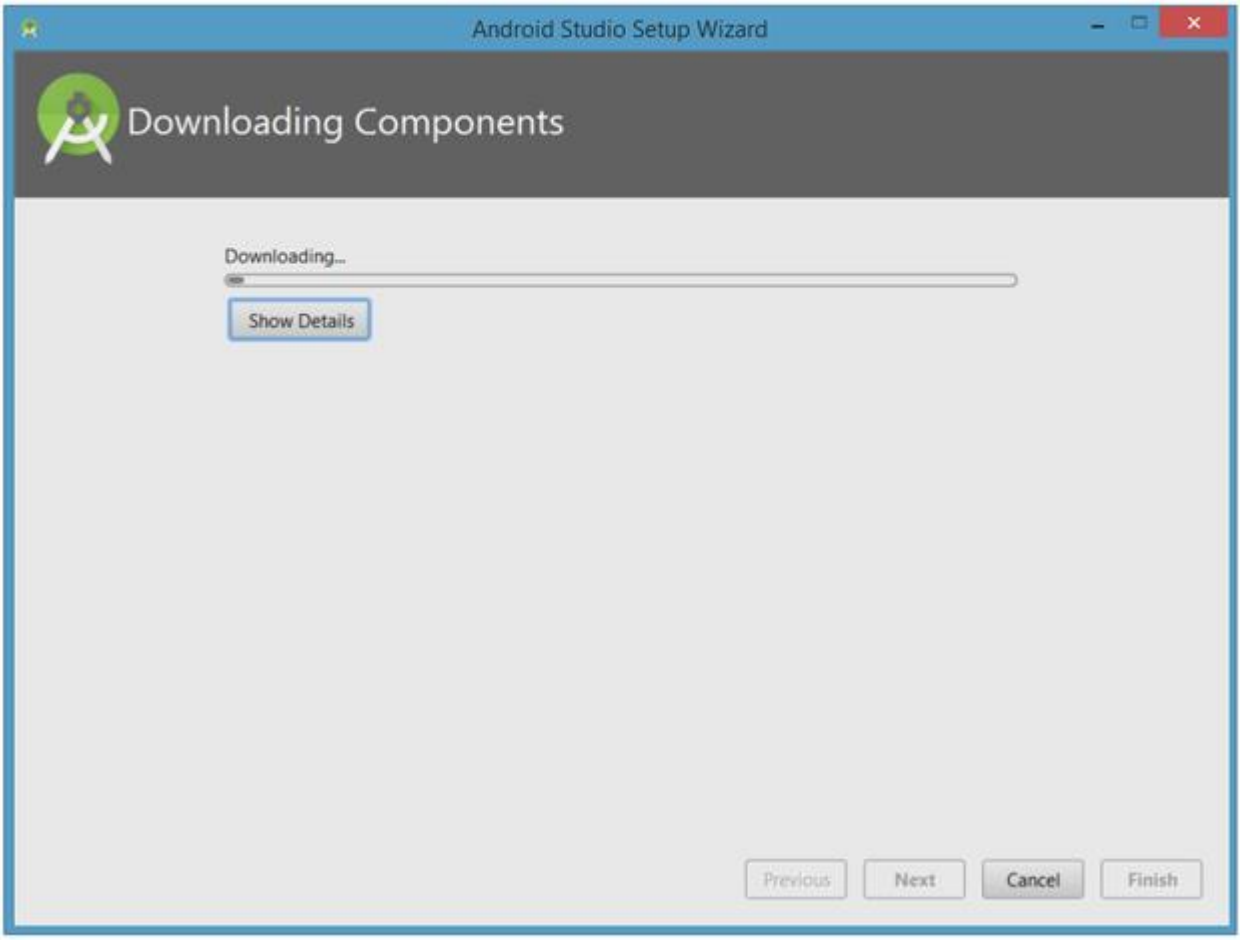

When you click next, the setup wizard invites you to select an installation type for your SDK components. For now I recommend you keep the default standard setting.




Click next and verify your settings, then click Finish to continue.




The wizard will download and unzip various components. Click Show Details if you want to see more information about the archives being downloaded and their contents.




If your computer isn't Intel based, you might get an unpleasant surprise after the components have completely downloaded and unzipped:

```
m2repository/com/google/firebase/firebase-core/9.0
 .1/firebase-core-9.0.1.pom.md5
Android SDK is up to date.
Unable to install Intel HAXM
Your CPU does not support required features (VT-x or SVM).
Unfortunately, your computer does not support hardware
 accelerated virtualization.
Here are some of your options:
 1) Use a physical device for testing
 2) Develop on a Windows/OSX computer with an Intel processor
 that supports VT-x and NX
 3) Develop on a Linux computer that supports VT-x or SVM
 4) Use an Android Virtual Device based on an ARM system image
   (This is 10x slower than hardware accelerated
 virtualization)

Creating Android virtual device
Unable to create a virtual device: Unable to create Android
 virtual device
```

Your options are to either put up with the slow emulator or use an Android device to speed up development. I'll discuss the latter option later in the tutorial.

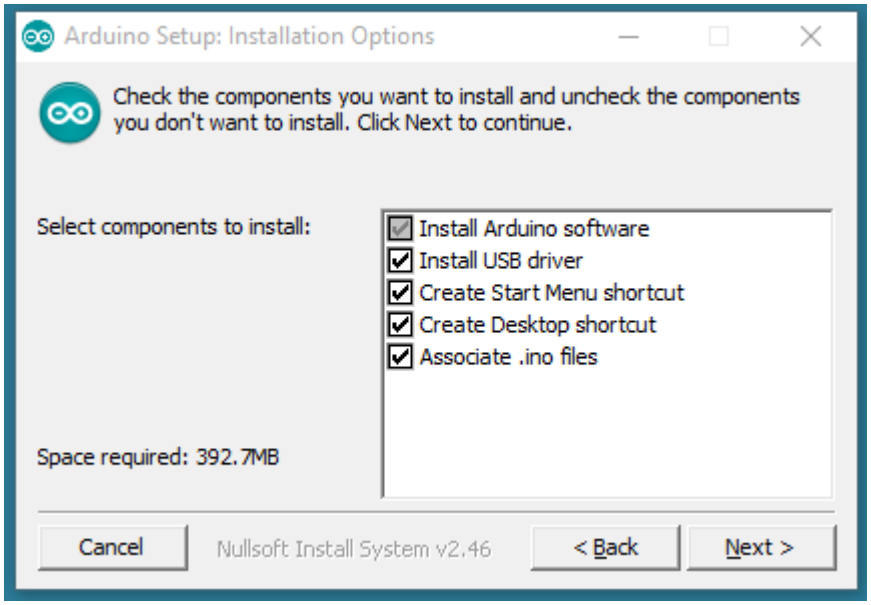
Finally, click Finish to complete the wizard. You should see the Welcome to Android Studio dialog box:

You'll use this dialog to start up a new Android Studio project, work with an existing project, and more. You can access it anytime by double-clicking the Android Studio shortcut on your desktop.

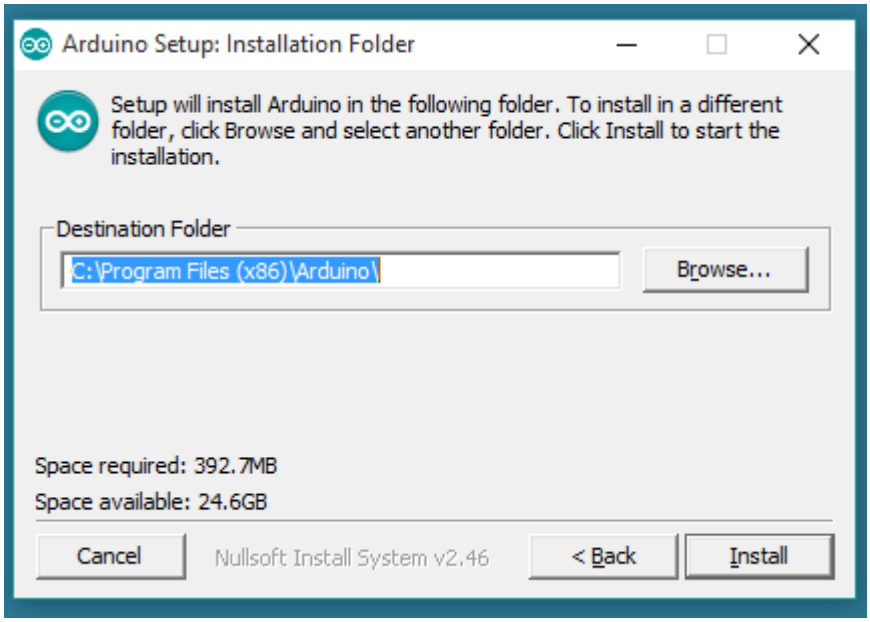
## Installation of Arduino:

Get the latest version from the download page. You can choose between the Installer (.exe) and the Zip packages. We suggest you use the first one that installs directly everything you need to use the Arduino Software (IDE), including the drivers. With the Zip package you need to install the drivers manually. The Zip file is also useful if you want to create aportable installation.

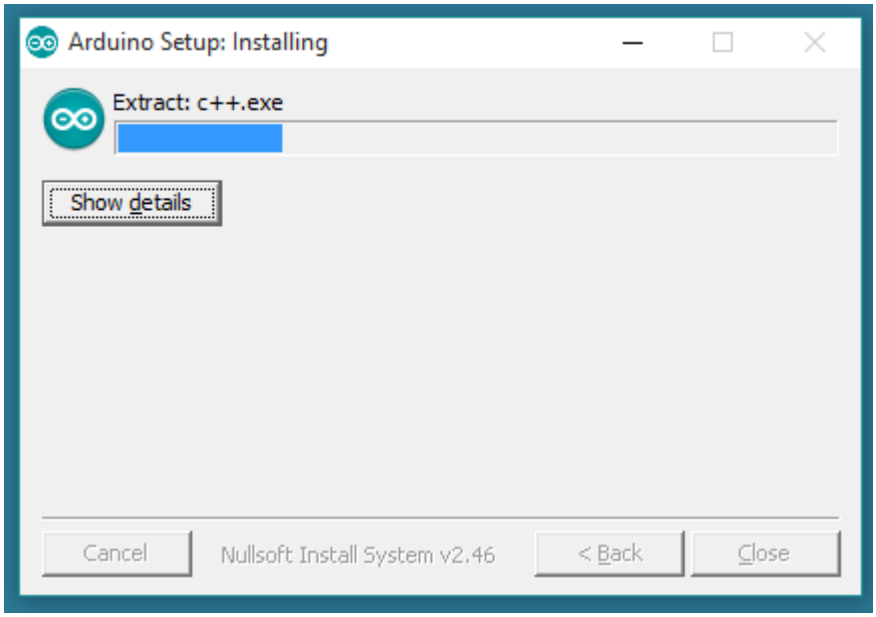
When the download finishes, proceed with the installation and please allow the driver installation process when you get a warning from the operating system.




Choose the components to install




Choose the installation directory (we suggest to keep the default one)




The process will extract and install all the required files to execute properly the Arduino Software (IDE)

## 4.5. Packages used for the system

***SmsManager to send SMS***

The SmsManager manages SMS operations such as sending data to the given mobile device. You can create this object by calling the static method SmsManager.getDefault() as follows:

SmsManager smsManager = SmsManager.getDefault();

Once you have SmsManager object, you can use *sendDataMessage()* method to send SMS at the specified mobile number as below:

smsManager.sendTextMessage("phoneNo", null, "SMS text", null, null);

Apart from the above method, there are few other important functions available in SmsManager class. These methods are listed below:

| S.N. | Method & Description |
|---|---|
| 1 | **ArrayList<String> divideMessage(String text)**<br>This method divides a message text into several fragments, none bigger than the maximum SMS message size. |
| 2 | **static SmsManager getDefault()**<br>This method is used to get the default instance of the SmsManager |
| 3 | **void sendDataMessage(String destinationAddress, String scAddress, short destinationPort, byte[] data, PendingIntent sentIntent, PendingIntent deliveryIntent)**<br>This method is used to send a data based SMS to a specific application port. |
| 4 | **void sendMultipartTextMessage(String destinationAddress, String scAddress, ArrayList<String> parts,** |

Table 4.1. Packages

***List of packages and classes used in android for our project:***

1. import android.Manifest:

   It names the java package for application. The package name serves as a unique identifier for the application.

2. import android.database.Cursor:

   This provide a read-Write access to the result set returned by the database query.

3. import android.database.sqlite.SQLiteException:

   It indicates that there was no error with SQLite parsing or execution.

4. import andoid.net.uri:

   URI refers to uniform resource identifier. This package is used to identify the resource in android.

5. Import android.os.Bundle:

   Bundle means a mapping from String keys to various parcelable values. This package is used for mapping from string keys to values of various types.

6. Import android.support.v4.app.ActivityCompat:

   This package helps for accessing the features of an Activity.

7. Import android.support.v4.content.ContextCompat:

   In android Context is a class which gives global information about an application environment.ContextCompact is a package used for accessing the features of a Context.

8. Import android.support.v4.app.AppCompatActivity:

   It is a base class for activities that use the support library action bar features.

9. Import android.Telephony.smsmanager:

This is a package which manages the SMS operations such as data, text and pdu SMS messages.

10. Import android.text.TextUtils:

    TextUtils are used to split the text according to the rules that are opaque to the user interface.

11. Import android.Util.Log:

    It is an API (Application Program Interface) for sending log output.

12. Import android.View.view:

    This package represents building block of User Interface .It is used to create methods for the User Interface View Group.

13. Import android.Widget.ArrayAdapter:

    It is a Concrete BaseAdapter that is backed by an array of arbitrary objects.

14. Import android.Widget.Button:

    This package is used to implement methods for Button.

15. Import android.Widget.EditText:

    It is a thin veneer over TextView that configures itself to be editable.Widget.EditText is a package used to implement methods on EditText.

16. Import android.Widget.ListView:

    This is a package used to implement methods on list of data stored in ListView.

17. Import android.Wideget.Textview:

    This is a package used to implement methods on TextView data.

18. Import android.Widget.Toast:

    A toast is a view containing a quick little message for the user. The toast class helps you create and show the message.

19. Import java.sql.Date:

    It is used to represents the date that can be stored in database.
20. Import java.text.SimpleDateFormat:

    It is a concrete class for formatting and parsing dates in locale- sensitive manner.

21. Import java.util.ArrayList:

    It provides methods to manipulate the size of the array that is internally stored to the list.

22. Import java.util.Calendar:

    It provides methods to manipulate the operations on date and time.

23. Import java.util.Date:

    Java provide the Date class available in java.util package, this class encapsulates the current date and time.

24. Import java.util,Locale:

    Locale is a mechanism for identifying objects.It performs an operation that requires a locale sensitive and uses the Locale to form information for the user.

25. Import java.util.concurrent.Timeunit:

    A TimeUnit is mainly used to inform time-based methods how a given timing parameter should be interpreted.

26. Import android.database.sqlite.SQLiteClosable:

    It exposes methods to manage a SQLite database.

27. Import android.database.sqlite.SQLiteOpenHelper:

    It is used to manage the database creation and version management.
28. Import android.content.ContentValues:

    This is used to store a set of values that the ContentResolver can process.

***List of packages used in android for our project:***

1. #include<dht.h>
   This is used to import the methods like humidity() and temperature() into the program.
2. #include<gsm.h>
   This is used to import the permissions required to access the messages from GSM module.
3. #include<SoftwareSerial.h>
   This is used to import the permissions to access the serial monitor for examine the output of the system.

## 4.6. Sample code

### *4.6.1.* Arduino DHT11 sensor:

```
intchk = DHT.read11(DHT11_PIN);
hum = DHT.humidity;
temp = DHT.temperature;
```

### *4.6.2.* Sending message from GSM900:

```
Serial.println("AT+CMGF=1");
delay(1000);
Serial.print("AT+CMGS=\"");
Serial.print(phone_no);
Serial.println("\"");
Serial.println("temperature");
Serial.println(temp);
Serial.println("humidity");
Serial.println(hum);
Serial.println((char)26);
delay(100);
Serial.write(0x1A);  // sends ctrl+z end of message
Serial.write(0x0D);  // Carriage Return in Hex
Serial.write(0x0A);  // Line feed in Hex
```

```
  //The 0D0A pair of characters is the signal for the end of a line and
beginning of another.
delay(5000);
```

### *4.6.3.* Receiving message from mobile to GSM900:

```
Serial.println("AT");
delay(1000);
Serial.println("ATE0");
delay(1000);
Serial.println("AT+CMGF=1");
delay(1000);
Serial.println("AT+CNMI=1,2,0,0");  delay(5000);
while(!Serial.available());

if(Serial.read() == '*')
   {
while(!Serial.available());
pass[0] = Serial.read();
while(!Serial.available());
pass[1] = Serial.read();
while(!Serial.available());
pass[2] = Serial.read();
while(!Serial.available());
pass[3] = 0;
tim = atoi(pass);
timsec = tim * 60;
delay(1000);
digitalWrite(led,HIGH);
while(timsec)
     {
timsec--;
```

```
        //if(timsec%2)
          //digitalWrite(led2,LOW);
         //else
         //digitalWrite(led2,HIGH);
delay(1000);
    }
digitalWrite(led,LOW);
  }
```

### *4.6.4.* XML Code for sending and receiving message

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayoutxmlns:android="http://schemas.android.com/apk/res
/android"
xmlns:tools="http://schemas.android.com/tools"
xmlns:app="http://schemas.android.com/apk/res-auto"
android:id="@+id/activity_main"
android:background="@mipmap/background"
android:layout_width="match_parent"
android:layout_height="match_parent"
tools:context="upgradekaro.smspreference.MainActivity">

<android.support.v7.widget.CardView
android:layout_width="match_parent"
app:cardElevation="20dp"
android:layout_margin="16dp"
app:cardCornerRadius="8dp"
app:cardBackgroundColor="#7effffff"
android:layout_height="match_parent" >

<LinearLayout
android:orientation="vertical"
```

```
android:layout_width="match_parent"
android:weightSum="2"
android:layout_height="match_parent"
android:layout_centerVertical="true"
android:layout_centerHorizontal="true">

<RelativeLayout
android:layout_width="match_parent"
android:layout_weight="1"
android:background="#852d9bb1"
android:layout_height="0dp">

<TextView
android:text="  -->  Data  From  Plant-->"
android:textColor="#fff"
android:layout_margin="5dp"
android:drawableRight="@mipmap/message"
android:drawableLeft="@mipmap/tree"
android:gravity="center"
android:layout_centerHorizontal="true"
android:drawableTint="#fff"
android:textSize="20dp"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:id="@+id/tvheadreceived" />

<ListView
android:layout_below="@+id/tvheadreceived"
android:layout_width="match_parent"
android:padding="5dp"
android:stackFromBottom="true"
android:id="@+id/lstview_recsms"
```

```
android:layout_height="match_parent" />
</RelativeLayout>

<RelativeLayout
android:layout_width="match_parent"
android:layout_weight="1"
android:layout_height="0dp">

<TextView
android:text=" -->   Commands to Plant -->”
android:textSize="20dp"
android:drawablePadding="2dp"
android:drawableLeft="@mipmap/message"
android:drawableRight="@mipmap/tree"
android:layout_centerHorizontal="true"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:id="@+id/tvsentmessagehead" />

<Button
android:text="Send"
android:layout_width="wrap_content"
android:layout_alignParentRight="true"
android:layout_alignParentBottom="true"
android:layout_height="wrap_content"
android:id="@+id/btn_send" />

<EditText
android:layout_margin="5dp"
android:layout_width="match_parent"
android:layout_height="45dp"
android:inputType="textPersonName"
```

```
android:hint="enter message"
android:ems="10"
android:textColorHint="#fff"
android:textColor="#fff"
android:background="@drawable/edmessage"
android:layout_toLeftOf="@+id/btn_send"
android:gravity="center"
android:layout_alignParentBottom="true"
android:layout_centerHorizontal="true"
android:id="@+id/edsendmessage" />

<ListView
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:id="@+id/sentsmslistview"
android:layout_alignParentRight="true"
android:layout_above="@+id/edsendmessage"
android:layout_alignParentTop="true"
android:layout_marginTop="20dp"
android:stackFromBottom="true"
android:layout_marginBottom="5dp"
android:layout_centerHorizontal="true"
android:layout_below="@id/tvsentmessagehead"
                          />
</RelativeLayout>
</LinearLayout>
</android.support.v7.widget.CardView>

</RelativeLayout>
```

### *4.6.5.* java code for sending and receiving message

```
import android.database.Cursor;
import android.database.sqlite.SQLiteException;
import android.net.Uri;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.telephony.SmsManager;
import android.text.TextUtils;
import android.util.Log;
import android.view.View;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.EditText;
import android.widget.ListView;
import android.widget.TextView;
import android.widget.Toast;

import java.util.ArrayList;

public class MainActivityextends AppCompatActivityimplements View.OnClickListener {
   String colName;
TextViewtView;
ListViewlstviewrec;
ListViewsentssmslistview;
Dbhelpredbhelpre;
   String phonenum="+919505628345";
// String  phonenum="9959582342";
Button send;
EditTextedsssendmessage;
ArrayListrecievedmessages=new ArrayList();
ArrayListSentsmslist=new ArrayList();
```

```
   @Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_main);
dbhelpre=new Dbhelpre(MainActivity.this);
send= (Button) findViewById(R.id.btn_send);
edsssendmessage= (EditText) findViewById(R.id.edsendmessage);
sentssmslistview= (ListView) findViewById(R.id.sentsmslistview);
lstviewrec= (ListView) findViewById(R.id.lstview_recsms);
send.setOnClickListener(this);
loadsentSmsList();
displaySmsLog();
   }
private void displaySmsLog() {
StringBuildersmsBuilder = new StringBuilder();
StringBuildersmssentBuilder=new StringBuilder();
final String SMS_URI_INBOX = "content://sms/inbox";
try {
        Uri uri = Uri.parse(SMS_URI_INBOX);
        String[] projection = new String[]{"_id", "address", "person", "body", "date", "type"};
        Cursor curread = getContentResolver().query(uri, projection,
"address='"+phonenum+"'",null, "date desc");
////read recieved messages /////
if (curread.moveToFirst()) {
intindex_Address = curread.getColumnIndex("address");
intindex_Body = curread.getColumnIndex("body");
do {
            String strAddress = curread.getString(index_Address);
            String strbody = curread.getString(index_Body);
//     smsBuilder.append(strAddress + ", \n");
```

```
                // smsBuilder.append(strbody + "\n\n");
recievedmessages.add(""+strbody);
Log.d("checkmessages",""+smsBuilder);
            }
while (curread.moveToNext());
if (!curread.isClosed()) {
curread.close();
curread = null;
            }
ArrayAdapterrecsmsAdapter=new
ArrayAdapter(MainActivity.this,R.layout.rrees,recievedmessages);
lstviewrec.setAdapter(recsmsAdapter);
            Toast.makeText(MainActivity.this,""+smsBuilder,Toast.LENGTH_SHORT).show();
        } else {
smsBuilder.append("no result!");
        } // end if
  ////// read sent messages///
} catch (SQLiteException ex) {
Log.d("SQLiteException", ex.getMessage());
      }
   }
public void sendSMS(String phoneNo, String msg) {
try {
SmsManagersmsManager = SmsManager.getDefault();
smsManager.sendTextMessage(phoneNo, null, msg, null, null);
Toast.makeText(getApplicationContext(), "Message Sent",
Toast.LENGTH_LONG).show();
      } catch (Exception ex) {
         Toast.makeText(getApplicationContext(),ex.getMessage().toString(),
Toast.LENGTH_LONG).show();
ex.printStackTrace();
```

```
        }
    }
    @Override
public void onClick(View v) {
if(v==send){
            String mess=edsssendmessage.getText().toString();
if(TextUtils.isEmpty(mess)){
edsssendmessage.setError("enter command");
            }
else {
// dbhelpre=new Dbhelpre(MainActivity.this);
edsssendmessage.setText("");
sendSMS(phonenum,mess);
dbhelpre.CreateTableForSent(mess);
loadsentSmsList();
            }
        }
    }
private void loadsentSmsList() {
ArrayAdapterarre=new ArrayAdapter(MainActivity.this,R.layout.sentitem,Sentsmslist);
arre.clear();
        Cursor c = dbhelpre.Readdatamessages();
if (c != null) {
if (c.moveToFirst()) {
do {
                    String name = c.getString(c.getColumnIndex("message"));
Sentsmslist.add(name);
sentssmslistview.setAdapter(arre);
                } while (c.moveToNext());
            } } }
```

# 5. TEST CASES

## 5.1 Introduction

A test case in software engineering is a set of conditions or variables under which a tester will determine whether an application or software system meets specifications. The mechanism for determining whether a software program or system has passed or failed such a test is known as a test oracle. In some settings an oracle could be a requirement or use case. It may take many test cases to determine that a software program or system is functioning correctly. Test cases are often referred to as test scripts, particularly when written. Written test cases are usually collected into test suites.

### 5.1.1 Levels of Testing

In order to uncover the errors present in different phases we have the concept of levels of testing. The basic levels of testing are

Client needs : Acceptance Testing

Requirements : System Testing

Design : Integration Testing

Code : Unit Testing

### 5.1.2 Software Testing Strategies

A strategy for software testing will begin in the following order.

- Unit Testing
- Integration Testing
- Validation Testing
- System Testing

#### 5.1.2.1 Unit Testing

It concentrates on each unit of the software as implemented in source code and is a white box oriented. Using the component level design description as a guide, important control paths are tested to uncover errors within the boundary of the module. In the unit testing, the steps can be conducted in parallel for multiple components. In my project I tested all the modules individually related to main function codes and attacks also.

#### 5.1.2.2 Integration Testing

Here focus is on design and construction of the software architecture. Integration Testing is a systematic technique for constructing the program structure while at the same time conducting tests to uncover errors associated with interfacing. The objective is to take unit tested components and build a program structure that has been dictated by design. The goal here is to see if modules can be integrated properly, the emphasis being on testing interfaces between modules.

#### 5.1.2.3 Validation Testing

In this, requirements established as part of software requirement analysis are validated against the software that has been constructed i.e., validation succeeds when software functions in a manner that can reasonably expected by the customer.

#### 5.1.2.4 System Testing

System testing of software or hardware is testing conducted on a complete, integrated system to evaluate the system's compliance with its specified requirements. System testing falls within the scope of black box testing, and as such, should require no knowledge of the inner design of the code or logic. Here the entire software system is tested. The reference document for this process is the requirements document, and the goal is to see if software meets its requirements.

## 5.2. Test cases

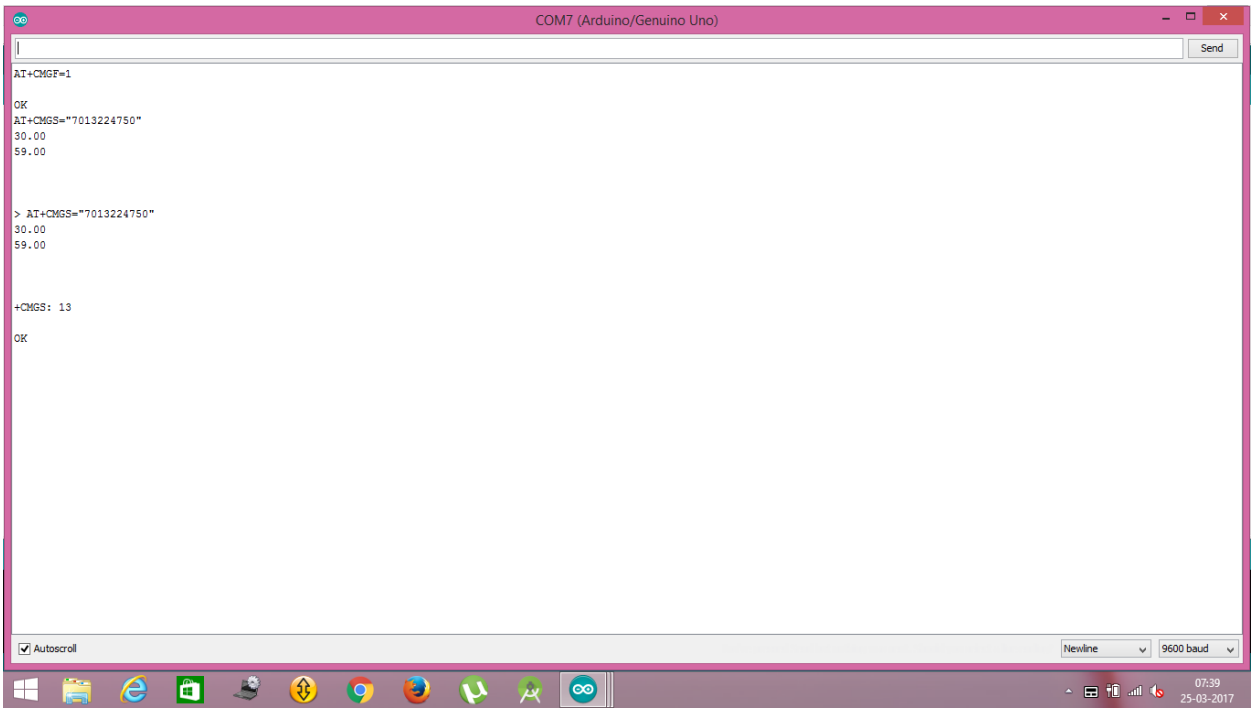
| Test case ID | Scenario | Input | Expected output |
|---|---|---|---|
| APWTC_01 | Receive message | Humidity: 31.2<br>Temperature:30.1 | Humidity: 31.2<br>Temperature:30.1 |
| APWTC_02 | Receive message | Humidity: 31.2<br>Temperature:30.1 | Humidity: 31.2<br>Temperature:30.1 |
| APWTC_03 | Send message | 95056***** | 95056***** |
| APWTC_04 | Send message | 95056***** | 95056***** |
| APWTC_05 | DHT11 sensor | Humidity: 31.2<br>Temperature:30.1 | Humidity: 31.2<br>Temperature:30.1 |
| APWTC_06 | DHT11 sensor | Humidity: 31.2<br>Temperature:30.1 | Humidity: 31.2<br>Temperature:30.1 |
| APWTC_07 | Mobile application number | Import from predefined number | Import from predefined number |

Table 5.1

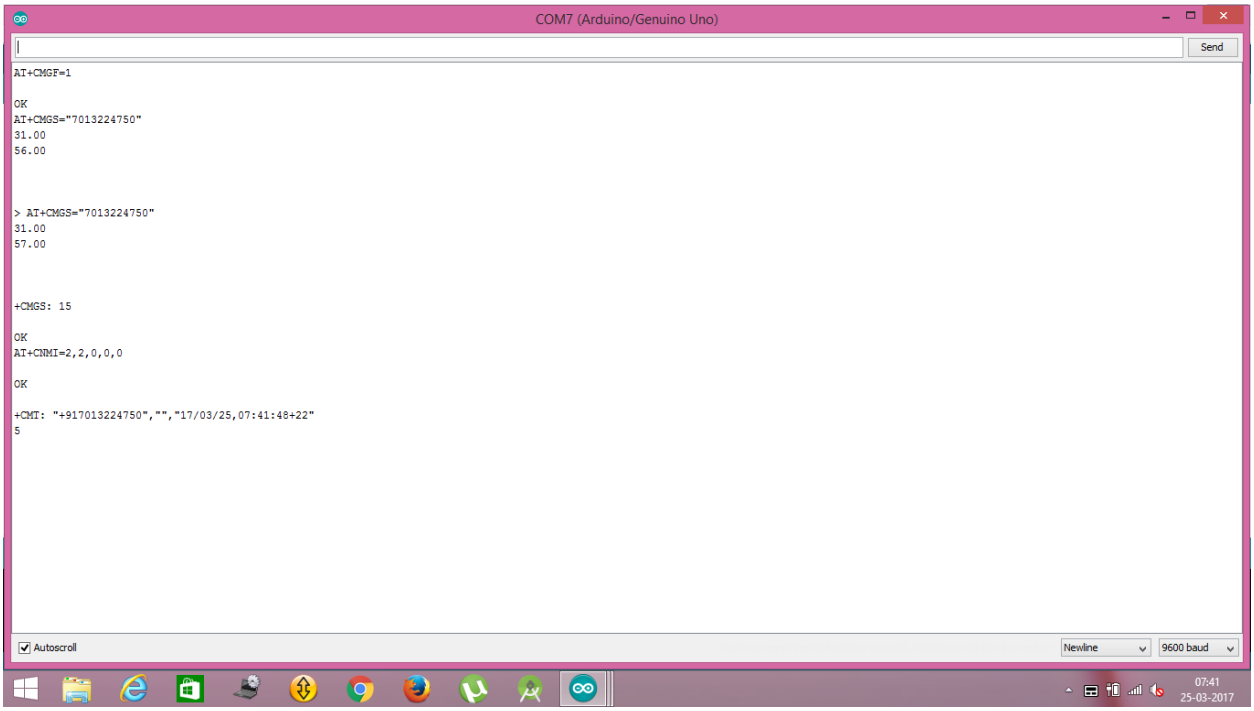
Test cases

# 6. SCREENS AND REPORTS

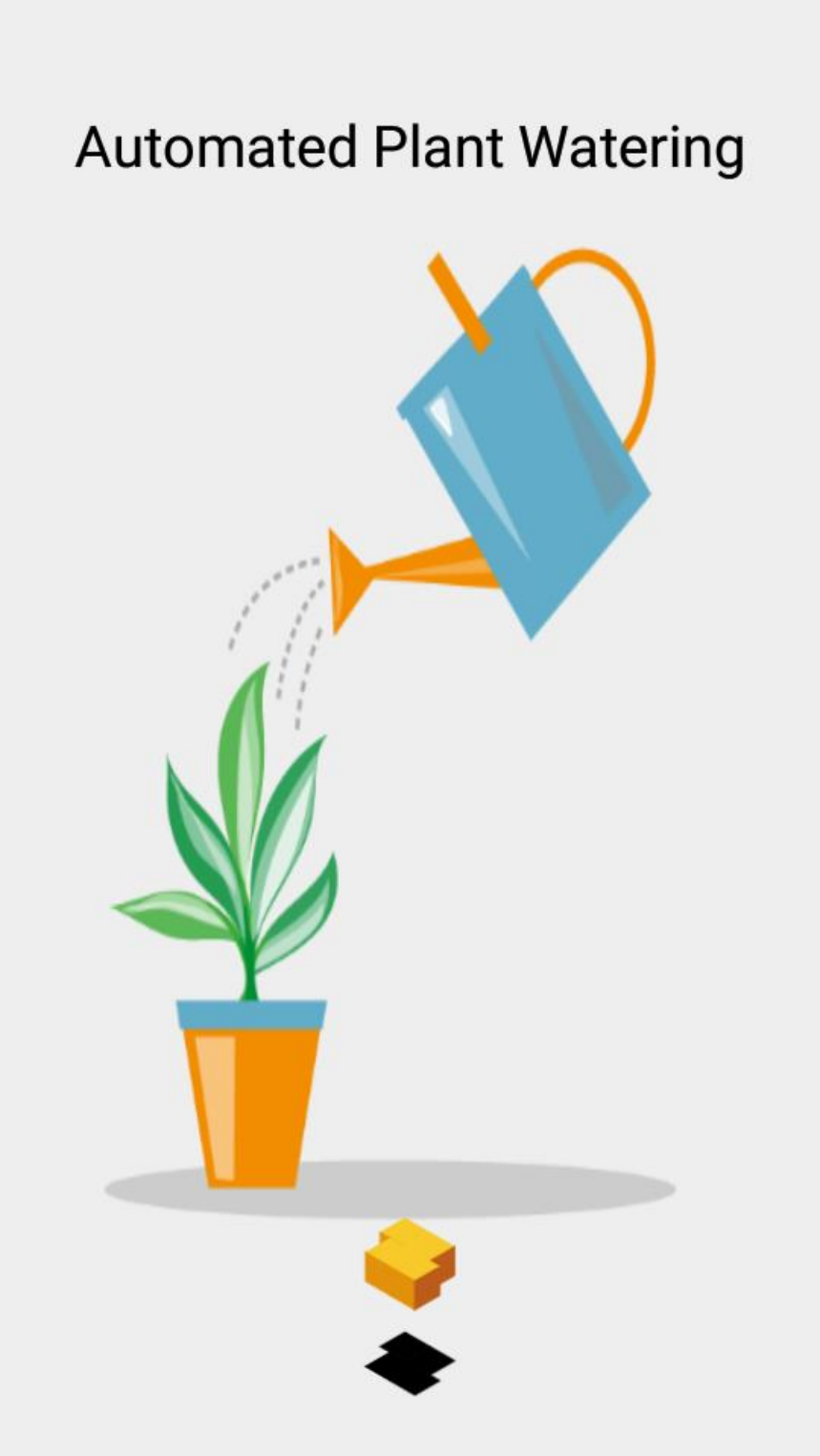
## 6.1. Screen shots



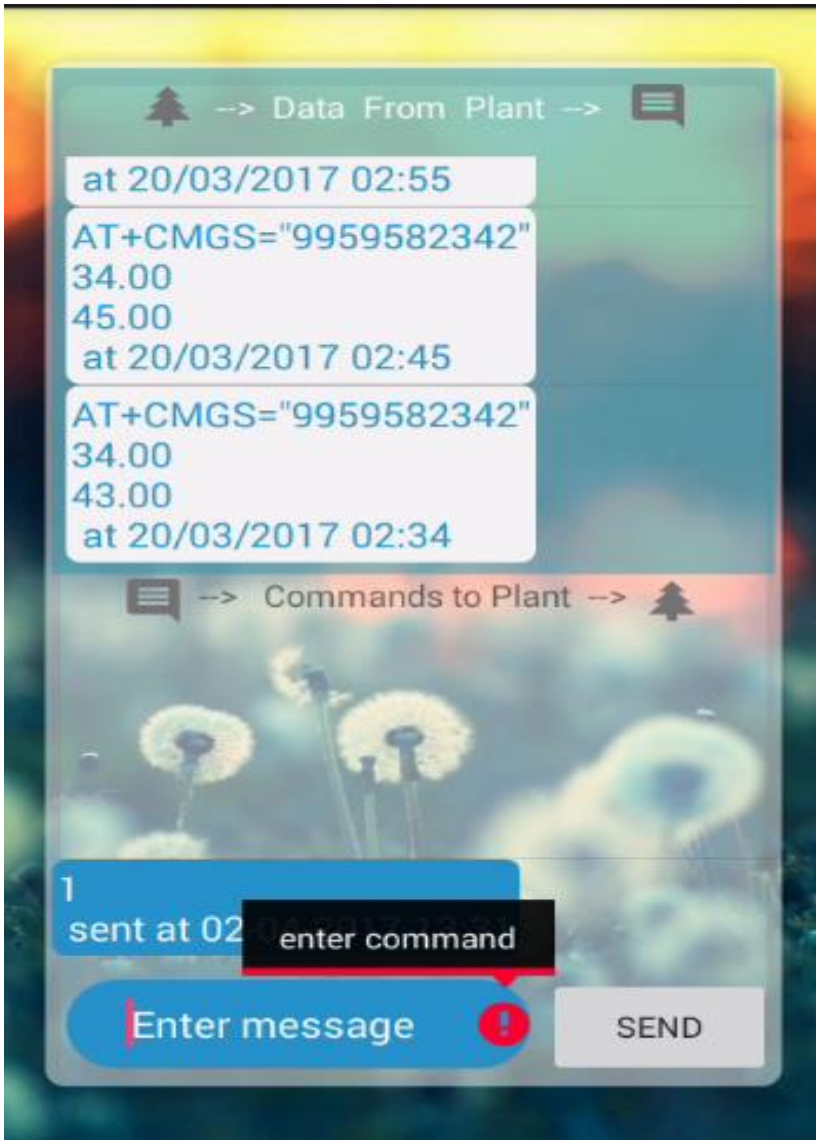

6.1. Screenshot of sending message



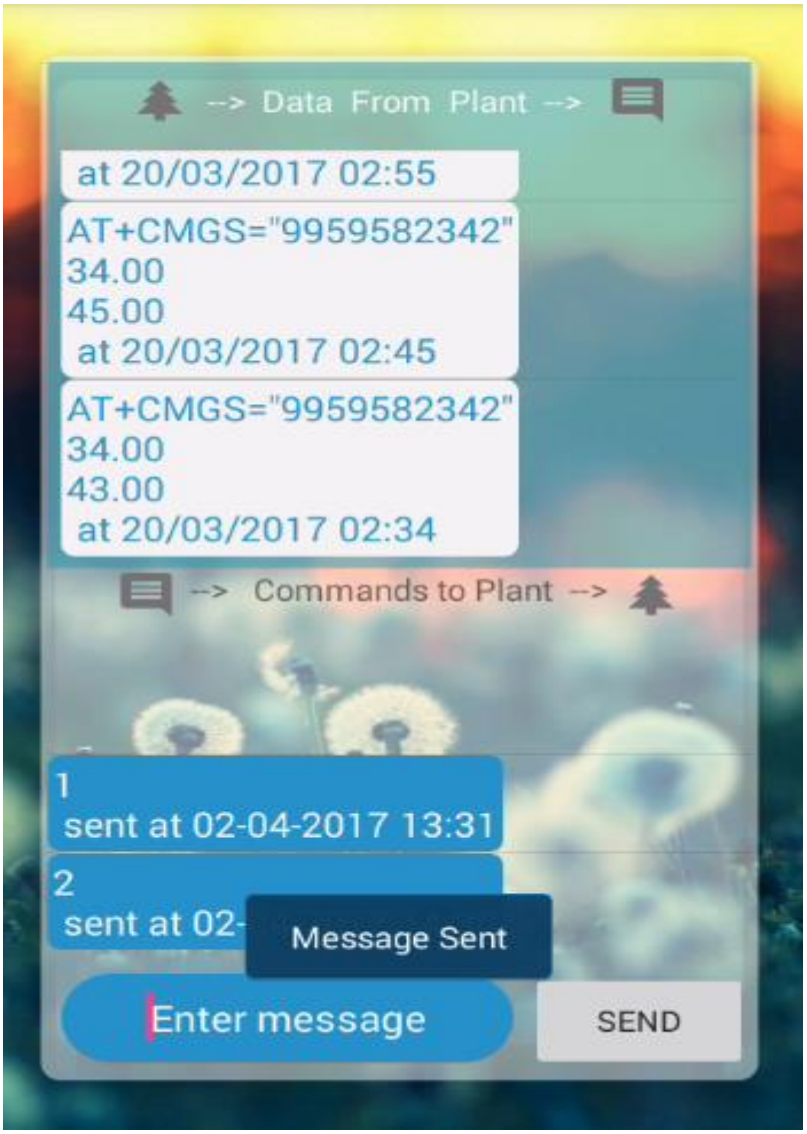

6.2. Screenshot of receiving message




6.3. Screenshot of splash screen




6.4. Screenshot of message display screen with error




6.5. Screenshot of toast message


6.6. Screenshot of sending message

## 6.2. Report

| Test case ID | Scenario | Input | Expected o/p | Actual o/p | Pass/fail |
|---|---|---|---|---|---|
| APWTC_01 | Receive message | Humidity:31.2 Temp:30.11 | Humidity:31.2 Temp:30.11 | Humidity:31.2 Temp:30.11 | Pass |
| APWTC_02 | Receive message | Humidity:31.2 Temp:30.11 | Humidity:31.2 Temp:30.11 | Humidity:0.00 Temp:0.00 | Fail |
| APWTC_03 | Send Message | 95056***** | 95056***** | 95056***** | Pass |
| APWTC_04 | Send Message | 95056***** | 95056***** | 99786***** | Fail |
| APWTC_05 | DHT11 Sensor | Humidity:31.2 Temp:30.11 | Humidity:31.2 Temp:30.11 | Humidity:31.2 Temp:30.11 | Pass |
| APWTC_06 | DHT11 Sensor | Humidity:31.2 Temp:30.11 | Humidity:31.2 Temp:30.11 | Humidity:0.00 Temp:0.00 | Fail |
| APWTC_07 | Mobile application number | Import from predefined number | Imported from predefined number | Imported from other number | Fail |

Table 6.1.Report on test cases

# 7. CONCLUSION AND FUTURE SCOPE

## 7.1. Conclusion

This is a simple watering system, which is very easy to use. The user can water the plant by just instructing the system through a simple text message from anywhere out of the home. So, this is a system which eases the gardener's task of watering a plant and ensures the healthy growth of the plant.

## 7.2. Future scope

Future scope of this project is to extend the functionalities like automatically shifting the control to the system without involving the human completely. The plant will be watered according to the conditions specified by the programmer previously.

# 8. BIBLIOGRAPHY

## Books referred


| sl.no | Book | Author | Publisher | Year |
|---|---|---|---|---|
| 1 | Head First Android | Dawn Griffiths | O'Reilly | 2003 |
| 2 | Android App Development For Dummies | Donn Felker with Joshua Dobbs | Wiley | 2011 |
| 3 | Arduino projects for dummies | Brook craft | Wiley | 2013 |


## Websites visited


- https://developer.android.com/index.html
- https://github.com/android
- http://stackoverflow.com/documentation/android/topics
- http://www.instructables.com
- http://www.arduino.cc