TABLE OF CONTENTS

Introduction

1.1. Purpose of Plan

As most of the work in this area has been done regarding the Arduino & its application, what left out most of the time is the Android controller. In this project we are trying to exploit the android open accessory Bundle so that we can give

a) More realistic experience to the user.

b) Better Connectivity to the Arduino Chip.

c) Increasing the efficiency in controlling of Bot.

1.2. Project objectives

In this project we will be constructing the android guided Arduino Car by extracting the powers of both the open source technologies- Android and Arduino Programming.

1.3. Project goals

- To develop an android application that will provide user an interface to interact with the Arduino powered car.
- To develop an appropriate program in the Arduino microchip to interact with
  the android controller.
- To compile all the developed modules that we constructed above.
- To produce Arduino car that is controlled by android phone remote which can be used in various fields, like defense, scientific expeditions and so on.

1.4. Scope Definition

The project is limited to designing an android interface, Arduino bot and write program in to the Arduino microprocessor. Arduino car contains Arduino microcontroller with basic mobility features. Arduino programs contains instructions mediating between android controller and Arduino car. Android mobile controller uses different mobile sensors to supervise motion.

Literature review

This literature review explores potential information to identify current knowledge and key issues relating to development of Android Supervised Arduino Car, which are divided into two sections: Arduino interface & programming, android application development.

2.1. Android [1][2][3][4]

Android was made up from the scratch by its founder who has been working on the Linux kernel. Their true aim was to design an operating system which is so powerful to use all its hardware & software power. They didn't expect that the openness concept of android is so much liked by the developer that android operating system has the largest app store than any other mobile operating system.

Android as we know a common term in the market of smartphone. A million of devices run on the android smartphone which is forwarded by the google in the wake for need of mobile operating system.

Google has made the Android an open source technology which is one of the reasons behind the success of Android. Google announced in its IO Conference the fusion of Android with Arduino which is also the open source technology.

As Google draw out more Arduino libraries for Android IDE which comes under the name of Android Open Accessory, it created numerous possibilities for developers to use Arduino side by side with Android & create application which can be used in various field.

Application anatomy

Running applications could be a major goal of in operation systems and android an operating system provides many suggests that on completely different layers to compose, execute and manage applications.

For this purpose, android an operating system clearly differentiates the terms application, process, task and thread. This chapter explains every term by itself yet because the correlation between the terms.

Processes & threads

Five varieties of processes area unit distinguished in android an operating system so as to manage the behavior of the system and its running programs. The varied varieties have completely different importance levels that are strictly ordered. The ensuing importance hierarchy for method categories feels like this (descending from highest importance,

- Foreground: A method that\'s running an Activity, a Service providing the Activity, a beginning or stopping Service or a presently receiving Broadcast Receiver.
- Visible: If a method holds a paused however still visible Activity or a Service absolute to a noticeable Activity and no foreground elements, it\'s classified a visible process.
- Service: A method that executes an already started Service.
- Background: associate degree Activity that\'s now not visible is hold by a background method.

- Empty: These processes contain no applications program elements and exists just for caching functions.

If the system is running low on memory, the importance of a method becomes an important half within the system's call that method gets killed to free memory. So empty processes are killed possibly followed by background processes then on. Typically, solely empty and background processes are killed that the user expertise stays unaffected. The system is intended to go away everything untouched as long as doable that\'s related to a noticeable part like associate degree Activity. Processes will contain multiple threads, love it is common on Linux primarily based systems. Most androidian operating system applications include multiple threads to separate the UI from input handling and I/O operations or long running calculations, thence the underlying processes are multi-threaded. The threads used on application level area unit commonplace Java threads running within the Dalvik VM

Applications & tasks

Android applications are surpassing processes and their enclosed threads. The 2 terms task and application are joined along tightly, only if a task may be seen as associate degree application by the user. In truth tasks are a series of activities of presumably multiple applications. Tasks primarily area unit a logical history of user actions, e.g. the user opens a mail application {in that during which within which} he opens a particular mail with a link enclosed which is opened during a browser. During this situation the task would come with 2 applications (mail and browser) whereupon there also are 2 Activity elements of the mail application and one from the browser enclosed within the task. A plus of the task conception is that the chance to permit the user to travel back step by step sort of a pop operation on a stack.

Application internals

The structure of associate degree golem application relies on four completely different elements that are: Activity, Service, Broadcast Receiver and Content Provider. Associate degree application doesn't essentially consist of all four of those elements, however to gift a graphical computer program there needs to be a minimum of associate degree Activity. Applications will begin alternative applications or specific elements of alternative applications by causing associate degree Intent. These intents contain among alternative things the name of desired dead action. The Intent Manager resolves incoming intents and starts the correct application or part. The reception of associate degree Intent may be filtered by associate degree application.

Services and broadcast receivers enable applications to perform jobs within the background and supply further practicality to alternative elements. Broadcast receivers may be triggered by events and solely run a brief amount of your time whereas a service could run a protracted time. The compiled code of the appliance elements and extra resources like libraries, pictures and alternative necessary information is packed into one .ask file that forms the feasible golem application.

AndroidManifest.xml

All androidian operating system Dalvik applications got to have an XML document within the application's root directory referred to as AndroidManifest.xml. This document is employed by numerous facilities within the system to get body and structure data concerning the appliance. Within the manifest file predefined part varieties area unit allowed to specify among alternative things the appliance name, the elements of the appliance, permissions, required libraries and filters for intents and broadcasts. Throughout development the manifest file holds the management data for instrumentation support. a number of the weather within the manifest file area unit mentioned in additional detail in alternative chapters, wherever it matches the context.

Activities

An Activity could be a single screen of associate degree application sort of a browser window or a settings page. It contains the visual components that gift information (like associate degree image) or enable user interaction (like a button). Every application will have multiple activities whereupon the transition between the various activities is initiated via intents. All activities area unit subclasses from android.app. Activity and their life cycle are controlled by the onXYZ () ways. This idea is required by Android's means of handling multitasking and helps coping with low memory things. The most functions are:

- on Create () The initial methodology to line up associate degree Activity.
- on Destroy () The counterpart to on Create ().
- onResume() This methodology is named if the Activity is visible within the
  foreground and prepared to induce and method user input.
- onPause() the strategy needs to quickly save uncommitted information and stop central processor intensive work to arrange the Activity to lose the main target and getting to background.
- onRestart() This methodology needs to restore an antecedently saved state of the Activity, because it is named once associate degree activity was utterly stopped and is required once more.

Intents, Intent filters and receivers

Unlike Content Providers, the opposite 3-part varieties of associate degree application (activities, broadcast receivers and services) area unit activated through intents. Associate degree Intent is associate degree asynchronously sent message object as well as the message that ought to be transported.

The contained message either holds the name of the action that ought to be performed, or the name of the action being declared. The previous applies to activities and services, the latter to broadcast receivers. The Intent category has some actions like ACTION EDIT and ACTION read or for broadcasts ACTION TIME TICK enclosed already. Additionally, to the action, the message contains a regular Resource symbol (URI) that specifies the info used for the given action. Optionally the Intent object will hold a class, a type, a part name, further information and flags.

Android utilizes completely different hooks within the application elements to deliver the intents. For associate degree activity, it's onNewIntent() methodology is named, at a service the onBind() methodology is named. Broadcast actions may be declared exploitation Context. sendBroadcast() or similar ways. Androidian operating system sends the Intent to the onReceive() methodology of all matching registered receivers.

Intents may be filtered by associate degree application to specify that intents may be processed by the application's elements. The list of filters is about within the application's manifest file, therefore androidian operating system will confirm the allowed intents before beginning associate degree application.

Content provider

The data storage and retrieval in androidian operating system applications is finished via content providers. These suppliers can even be wont to share knowledge between multiple applications, providing the concerned applications own the right permissions to access the information. Androidian operating system already has default suppliers for e.g. images, videos, contacts and settings which may be found within the golem. Provider package. AN application queries a Content Resolver that returns the suitable Content Provider. All suppliers square measure accessed like databases with AN URI to work out the supplier, a field name and also the knowledge style of this field. Applications solely access content suppliers via a Content Resolver, ne'er directly. If AN application desires to store knowledge that doesn't have to be compelled to be shared, it will use a neighborhood SQLite Database.

Background activities

Applications may have to perform some supporting operations within the background or while not a graphical interface in the least. Androidian operating system provides the 2 categories Broadcast Receiver and repair for these functions. If solely a brief operation needs to be performed, the Broadcast Receiver is most well-liked and for long running jobs a Service is most well-liked. Each category doesn’t essentially imply that the background part runs as a thread or perhaps in its own method, thence androidian operating system doesn't enforce this sort of behavior. so as to not freeze a running application, the service or broadcast receiver sometimes is running in its own thread, otherwise androidian operating system tends to kill such AN application because it appears not responsive. The Broadcast Receiver is activated through the onReceive () technique and gets nullified on come back from this technique. This makes it necessary to solely use synchronous ways in a very receiver. A broadcast commonly is unordered and sent to any or all matching receivers at a similar time, however it may be ordered too. During this case androidian operating system sends the printed solely to the one receiver at a time. This receiver runs as was common {and will and may and might} forward a result to future receiver or it can even abort the entire broadcast. A Service permits AN application to perform long running tasks in background and supply a number of the application's practicality to alternative applications within the system. A service may be utilized in 2 ways that, it's either started with one command or started and controlled by an incoming association that creates use of Remote Procedure Calls. each receiver and repair have to be compelled to be declared within the application manifest file to permit androidian operating system to work out the service's or receiver's category although the applying isn't running.

Application lifetime & states

The state androidian operating system application is in, is decided by the state of its parts, most significantly its activities. Because the application parts alter their states, the application's underlying method sort is adjusted. On application begin the individual parts start and just in case of an activity the subsequent hooks square measure referred to as sequential: onCreate(), onStart(), onResume(). The primary hook is just referred to as once in AN activities time period, however the opposite 2 ways will get referred to as additional typically. If AN activity loses the main focus, the onPause() technique is termed and if the activity isn't any longer visible, onStop() is termed. Before deleting AN activity, it's onDestroy() technique is run that ends the activity time period. Every technique gets referred to as on a special event to permit the activity to preserve its state or begin and restart properly. The subsequent list describes the aim of those hooks and the way the inflicting events amendment the applying state.

onCreate() This technique is termed for initialization and static originated functions. it should get passed AN older state for resuming. Future technique is often onStart(). onRestart() when AN activity is stopped and near to be started once more, this hook is termed and when it onStart(). onStart() the applying method sort changes to visible and also the activity is near to be visible to the user, however it’s not within the foreground.

onResume() The activity has the main focus and may get user input. The applying method sort is about to foreground.

onPause() If the applying loses the main focus or the device goes to sleep, this hook is termed and also the method sort is about to visible. When running this hook, the system is allowed to kill the applying at any time. All CPU intense operations ought to be stopped and doomed knowledge ought to be saved. The activity could also be resumed or get stopped.

onStop() The activity isn't any longer visible, the method sort is about to background and also the application could also be killed at any time by the system to regain memory. The activity is either about to get destroyed, or restarted.

onDestroy() The last technique that's referred to as in AN activity right before the system kills the applying or the applying deletes the activity. The applying method could also be set to empty if the system keeps the method for caching functions.

For a service, the time period is easier than the one among AN activity, because the onResume(), onPause() and onStop() hooks don't exist. For interactive services, there square measure the onBind(), onUnbind() and onRebind() hooks that square measure referred to as to begin, stop and restart a service. The method sort alters between foreground at creation and deletion time and repair at run time. Broadcast receivers solely have the onReceive() hook that runs at foreground method importance If AN activity has to save its state to gift the user a similar actual state the activity was in once it had been left, the onSaveInstanceState() technique may be used for this purpose. This hook is termed before the onPause() hook is termed and permits to save lots of the dynamic knowledge as key-value pairs in a very Bundle. The saved state object may be passed to onCreate() and onRestoreInstanceState() to revive the state of the activity.

RPC

Android includes a Common Object Request Broker design (CORBA) and part Object Model (COM) like light-weight RPC mechanism and brings its own language, AIDL (Android Interface Definition Language). AIDL uses proxy categories to pass messages between the server and also the shopper. The aidl tool creates Java interfaces from the interface definition that have to be compelled to be obtainable at the shopper furthermore as at the server. The created interface has AN abstract inner category referred to as Stub that needs to be extended and enforced by the shopper and also the server. If the server provides a Service, it's to come back AN instance of the interface implementation at its onBind() technique. Solely ways square measure allowed in AIDL and every one of them square measure synchronous. The mechanism is supported by the kernel's binder driver.

Application security

The security model of golem heavily depends on the multi-user capabilities of the underlying Linux. Every application runs with its own distinctive user id and in its own method. All Dalvik applications run in a very sandbox that by default prohibits e.g. communication with alternative processes, access to others knowledge, access to hardware options like GPS or camera and network access. Opposing to platforms with native binary executables, androidian operating system makes it straightforward to enforce a particular application behavior, as its application VM Dalvik directly controls code execution and resource access. Platforms like iOS, webOS, Symbian or MeeGo don't have this chance, therefore their sandboxing systems square measure supported means that on kernel-, filesystem- or process-level and a few of those means that square measure utilized by androidian operating system too.

The basic sandbox denies all requests from AN application unless the permissions square measure expressly granted. The permissions square measure originated within the application manifest file with the tag. That enables the system to raise the user or a package manager direct at install time for the application's needed permissions. Once put in, AN application will assume that everyone needed permissions square measure granted.

During the installation method, AN application is appointed with a novel and permanent user id. This user id is employed to enforce permissions on method and classification system level. AN application will expressly share files by setting the file mode to be world legible or writeable, otherwise all files square measure non-public. If 2 applications ought to share a similar process or use a similar user id, each of the applications have to be compelled to be signed with a similar certificate and invite a similar shared User Id in their manifest files.

The individual parts of AN application can even be restricted, to confirm that solely bound sources square measure allowed to begin AN activity, a service or send a broadcast to the application's receiver. A service will enforce fine grained permissions with the Context.checkCallingPermission() technique. Content suppliers will prohibit overall scan and write access furthermore as grant permissions on an URI basis.

Native applications

Android applications that require additional performance than the Dalvik VM offers, may be divided. One half stays within the Dalvik VM to produce the applying UI and a few logics and also the alternative half runs as native code. This manner applications will profit of the device capabilities, even though robot or Dalvik don\'t provide a particular feature. The native code components of Associate in an application are shared libraries that are referred to as through the Java Native Interface (JNI). The shared library has got to be enclosed within the applications .apk file and expressly loaded. The code from the native library is loaded into the address house of the application's
VM. This results in an attainable security hole, as of revision four of the Native Development Kit (NDK), the code will build use of the ARMv5TE and ARMv7-A directions sets. Additionally, the Vector Floating purpose (VFP) and atomic number 10 (Single Instruction Multiple knowledge instructions) extensions may be utilized in ARMv7-A code. Code utilized by native applications ought to solely build use of a restricted library set that amongst others includes the libc, libm, libz and a few 2nd and 3D graphics libraries. The performance speeding of native code over Dalvik taken code was examined. The results show that there\'s a large distinction between native and taken code, however the benchmarks were run on Associate in Nursing early androidian operating system version that e.g. didn't\'t provide a simply in time compiler.

Dalvik VM

Android applications and also the underlying frameworks are nearly entirely written in Java. Rather than employing a normal Java virtual

machine, robot uses its own VM. This virtual machine is\'t compatible to the quality Java virtual machine Java me because it is specialized and optimized for tiny systems. These little systems sometimes solely give very little RAM, a slow central processing unit and aside from most PCs no swap file to compensate the little quantity of memory. At now androidian operating system tells itself aside from alternative mobile operative systems like Symbian, Apple's iOS or Palm's webOS that use native compiled application code. The most programming languages used there are C, C++ and objective C, whereat e.g. internet OS permits alternative largely web primarily based languages like JavaScript and hypertext markup language further. The mandatory computer memory unit code interpreter – the virtual machine – is termed Dalvik .Instead of mistreatment normal computer memory unit code, Dalvik has its own computer memory unit code format that is adjusted to the requirements of robot target devices. The computer memory unit code is additional compact than usual Java computer memory unit code and also the generated .dex files
are little.

Design needs

As a multitasking software system, androidian operating system permits each application to be multithreaded and additionally to be touch multiple processes. For the sake of improved stability and increased security every application is separated from alternative running applications. Each application runs in exceedingly sandboxed surroundings in its own Dalvik virtual machine instance. This needs Dalvik to be little and solely add very little overhead.

Dalvik is intended to run on devices with a minimum total memory of simply sixty-four MB of RAM. For higher performance actual devices have quite sixty-four MB put in. of those sixty-four MB solely regarding forty MB stay for applications, libraries and services. The used libraries are quite giant and sure want ten MB of RAM. Actual applications solely have around twenty MB left of the sixty-four MB of RAM. This terribly restricted quantity of memory has got to be used expeditiously so as to run multiple applications right away.

There are 2 major areas to think about for minimizing the memory usage. First the applying itself has got to be as little as attainable and second the memory allocation of every application has got to be optimized. Additionally, to the reduced usage of valuable memory one gains quicker application load times and fewer required disk storage. Most significantly the ability offers by battery and also the used CPUs confine the allowed and attainable overhead for Dalvik. The generally used ARM CPUs solely give fairly restricted process power and little caches.

General & file optimizations

Java applications for Dalvik get compiled like alternative Java programs with constant compilers and largely constant toolchain. Rather than compression and packaging the ensuing category files into a .jar file, they\'re translated into .dex files by the dx tool. These files embody the Dalvik computer memory unit code of all Java categories of the applying. At the side of alternative resources like pictures, sound files or libraries the .dex files are packaged into .apk files. So as to save lots of space for storing, .dex files solely contain distinctive knowledge. If multiple category files share constant string, this string would solely exist once within the .dex file and also the multiple occurrences are simply tipping to this one string. Constant mechanism is employed for technique names, constants and objects which ends up in smaller files with a lot of internal “pointing”.

Byte code format

The Dalvik computer memory unit code is intended to cut back the amount of necessary memory reads and writes and inflated code density compared to Java computer memory unit code. For this purpose, Dalvik uses its own instruction set with a set instruction length of sixteen bit. Moreover, Dalvik could be a register primarily based virtual machine that among alternative things reduces the required code units and directions. The given register dimension is {32|thirty 2} bit that permits every register to carry two directions, sixty-four-bit values are hold in adjacent registers. Directions shorter than sixteen bit zero-fill the unused bits and pad to sixteen bits.

Dalvik is aware of a pair of56 completely different op codes whereof twenty-three are unused in robot 2.2, resulting in an actual total op code variety of 233. Optimizations of the computer memory unit code is generally done by the depot tool at installation time.

Install time work

For the beginning of an application, the associated .dex file has a minimum of to be verified for structural integrity. This verification will take a while and is required just one occasion, as application files don't amendment unless the applying is updated or uninstalled. As a result of that, the verification method may be done either at the primary startup or throughout the installation or update method. In androidian operating system this verification method is completed at installation time, in order that the least bit application startups later solely e.g. a check of the .dex file is required to verify it.

Verification

The verification and improvement procedure are performed by the dexopt program. To verify a .dex file, dexopt hundreds all categories of the file into a concisely initialized VM and runs through all directions all told strategies of every category. This manner amerceable directions or instruction mixtures may be found before the applying truly runs for the primary time. Categories during which the verification method succeeded get marked by a flag to avoid rechecking this category once more. So as to visualize the .dex files integrity, a CRC-32 check is kept among the file.

Optimization

After the successful verification of a .dex file, it gets optimized by dexopt. The optimizations aim at performance increase through reduced code size or reduced code complexness. The improvement mechanisms heavily rely upon the target VM version and also the host platform that makes it laborious to run dexopt elsewhere than on the host. The ensuing code is unoptimized Dalvik computer memory unit code mixed with optimized code mistreatment op codes not outlined within the Dalvik computer memory unit code specification. If necessary for the processor design endianness, the code is computer memory unit swapped and aligned consequently.

For code reduction dexopt prunes all empty strategies and replaces them with a no operation op code. Inclining some fairly often referred to as strategies like String. Length() reduces {the technique the tactic the strategy} decision overhead and virtual method decision indices get replaced by table indices. Moreover, field indices get replaced by computer memory unit offsets and short knowledge varieties like char, computer memory unit and Boolean are incorporate into thirty-two-bit kind to save lots of valuable central processing unit cache house. If it\'s attainable to pre-compute .dex file knowledge, dexopt appends the ensuing knowledge that reduces the central processing unit time required by the executing VM later. Each verification and improvement are restricted to method only 1 .dex file at a time that results in issues at handling dependencies. If a .dex file is optimized, it contains a listing of dependencies which may be found within the bootstrap category path. So as to ensure consistency just in case of changed .dex files, solely dependencies to .dex files within the bootstrap category path are allowed. this manner the verification of strategies looking on external .dex files aside from those in bootstrap can fail and also the connected category won\'t be optimized.

Optimizations of memory allocation and usage

In Dalvik there are a unit four completely different forms of memory to differentiate which will be sorted to clean/dirty and shared/private. Typical knowledge residing in either shared or non-public clean memory area unit libraries or application specific files like .dex files. Clean memory is protected by files or alternative sources and might be cropped by the kernel while not knowledge loss. The non-public dirty memory typically consists of the applications heap and writeable management knowledge structures like those required in .dex files. These 3 classes of various memory area unit quite common and no specialty of Dalvik.

Zygote

Shared dirty memory is feasible through a facility of Dalvik known as fertilized ovum. It's a process that starts at boot time and is that the parent of all Dalvik VMs within the system. The zygote loads and initializes categories that area unit speculated to be used fairly often by applications into its heap. In shared dirty memory resides e.g. the dex knowledge structures of libraries. When the startup of the zygote, it listens to commands on a socket. If a brand-new application starts, a command is distributed to the zygote that performs a typical fork(). The fresh forked method becomes a full Dalvik VM running the started application. The shared dirty memory is “copy-on-write” memory to attenuate the memory consumption.

Garbage assortment

The garbage collector (GC) in automaton runs in every VM on an individual basis, so every VMs heap is garbage collected severally. The zygote method and also the idea of shared dirty memory needs the gigahertz knowledge structures (“mark bits”) to not be tied to the objects on the heap, however unbroken separate. If the mark bits would lie next to the objects on the heap, a run of the gigahertz would bite these bits and switch the shared dirty memory into non-public dirty memory. So as to attenuate the nonpublic dirty memory the required mark bits area unit allotted simply before a gigahertz run and freed afterward.

JIT

Dalvik was designed to be an easy interpreter while not the aptitude to perform simply In Time (JIT) compilations. Android 1.0 was proclaimed expressly while not a JIT, as a result of it absolutely was seen as being too memory overwhelming and with a comparatively little performance boost. With the recent unleash of automaton two.2 the platform got a JIT for ARM processors that ought to have a coffee memory footprint and supply a notable performance boost. Androidian operating system applications usually decision native compiled and optimized libraries to perform performance essential tasks that results in solely concerning 1/3 of all dead code to be understood by the virtual machine. So, the doable quickening is restricted and conjointly depends heavily on the benchmarked application.

Types of JITs & Android’s JIT

In the wide field of simply in time compilers the strategy primarily {based} and also the trace based JITs area unit particularly attention-grabbing to androidian operating system. Methodology primarily {based} JITs compile an entire methodology to native code whereas trace based JITs solely compile one decision path. The benefits and drawbacks of those 2 compiler classes area unit listed below:

- Method based mostly JIT
- Large improvement window.
- Easy to adjust JIT and interpreter at methodology boundaries.
- Waste of area and time on code that's not run usually or not run the least bit.
- High memory usage for compilation.
- Takes lasting to learn.
- Trace based mostly JIT
- No methodology boundaries for improvement.
- Only “hot” code is compiled.
- At exceptions simply, rollback and begin interpreter.
- Limited improvement window of trace.
- Much overhead if the transition between JIT and interpreter is pricey.
- Sharing code between CPUs or Threads is tough.

Android’s JIT

Due to memory constrains the JIT utilized in automaton is trace based mostly and also the trace length is barely a hundred op codes short. The JIT and also the ancient interpreter area unit tightly intermeshed. just in case of an exception within the code from the JIT, the interpreter takes over, resets the state to the start of the trace and runs the code sequence once more while not the JIT. This mechanism is extended to permit parallel execution of JIT code and understood code to verify the

JIT against the interpreter. The androidian operating system JIT uses trace caches per method in order that threads within the same application share the trace cache. The cache size is configurable throughout build time whereas its default is concerning a hundred kb. So as to scale back overhead and increase performance, the JIT is ready to chain consecutive traces to avoid the requirement of change back to the interpreter. The code generated by the JIT is either traditional ARM code, Thumb or Thumb-2 code and in special cases the ARM VFP extensions area unit used for floating purpose operations. The code model utilized by the JIT is organized at build time and is adjusted to the capabilities of the target device central processing unit. The Thumb-2EE directions don't seem to be used despite the fact that they're designed to accelerate code generated by JITs. Androidian operating system takes advantage of interleaving and rearranging computer memory unit code directions to optimize the performance that contradicts Thumb-2EE’s idea of fast simply single directions.

Power management & kernel

The kernel utilized in androidian operating system may be a two.6 series UNIX kernel changed to meet some special wants of the platform. The kernel is generally extended by drivers, power management facilities and changes to the restricted capabilities of Android’s target platforms. The facility management capabilities area unit crucial on mobile devices, so the foremost necessary changes is found during this space. Just like the remainder of androidian operating system, the kernel is freely accessible and also the development method is visible through the general public androidian operating system supply repository. There are a unit multiple kernels accessible within the repository like an architecture general common kernel Associate in an experimental kernel. Some hardware specific kernels for platforms like MSM7xxx, Open multimedia system Application Platform (OMAP) and Tegra exist within the repository too.

Differences to mainland

The changes to the inject kernel is classified into: bug fixes, facilities to boost user area (lowmemorykiller, binder, ashmem, logger, etc.), new infrastructure (esp. wake locks) and support for brand new SoCs (msm7k, msm8k, etc.) and boards/devices. The androidian operating system specific kernels and also the inject UNIX kernel area unit speculated to get incorporate within the future, however this method is slow and can take your time.

Wake locks

Android permits user area applications and so applications running within the Dalvik VM to forestall the system from coming into a sleep or suspend state. this can be necessary as a result of by default, androidian operating system tries to to place the system into a sleep or higher a suspend mode as presently as potential. Applications will assure e.g. that the screen stays on or the central processing unit stays alert to react quickly to interrupts. This suggests that androidian operating system provides for this task area unit wake locks. Wake locks is obtained by kernel parts or by user area processes. The user area interface to form a wake lock is that the file /sys/power/wake lock within which the name of the new wake lock is written. To unleash a wake lock, the holding method writes the name in /sys/power/wake unlock. The wake lock is supplied with a timeout to specify the time till the wake lock are going to be discharged mechanically. All by the system presently used wake locks area unit listed in /proc/wake locks. The kernel interface for wake locks permits to specify whether or not the wake lock ought to stop low power states or system suspend. A wake lock is made by wake lock init() and deleted by wake lock destroy(). The created wake lock is uninheritable with wake lock() and discharged with wake unlock(). Like in user area it's potential to outline a timeout for a wake lock. The idea of wake locks is deeply integrated into androidian operating system as drivers and lots of applications create significant use of them. This can be a large obstacle for the androidian operating system kernel code to induce incorporate into the UNIX inject code.

Power manager

The Power Manager may be a service category within the application framework that offers Dalvik VM applications access to the Wake Lock capabilities of the kernel power management driver. Four completely different forms of wake locks area unit provided by the Power-

Manager:

PARTIAL_WAKE_LOCK: The central processing unit stays awake, albeit the device's power button is ironed

SCREEN_DIM_WAKE_LOCK: The screen stays on, however is dim

SCREEN_BRIGHT_WAKE_LOCK: The screen stays on with traditional brightness

FULL_WAKE_LOCK: Keyboard and screen continue with traditional back lightweight

Only the PARTIAL_WAKE_LOCK assures that the central processing unit is totally on, the opposite 3 sorts permit the central processing unit to sleep when the device's power button is ironed. Like kernel area wake locks, the locks provided by the Power Manager is combined with a timeout. It's conjointly potential to rouse the device once the wake lock is nonheritable or activate the screen at unleash.

Memory management

The memory management connected changes of the kernel aim for improved memory usage in systems with a little quantity of RAM. Each ashmem and pmem add a replacement method of allocating memory to the kernel. Ashmem may be used for allocations of shared memory and pmem permits allocations of contiguous memory.

ASHMEM: The Anonymous/Android Shared Memory provides a named memory
block that may be shared across multiple processes. Apart from the same old shared memory, the anonymous shared memory may be freed by the kernel. To use ashmem, a method opens /dev/ashmem and performs mmap() on that.

PMEM: Physical Memory permits e.g. drivers and libraries to allot named physically contiguous memory blocks. This driver was written to compensate hardware limitations of a particular SoC – the MSM7201A.

Low memory killer: the quality UNIX operating system kernel out of memory killer (oom killer) utilizes heuristics and figure {the method the method}' "badness" to be ready to terminate the process with the best score in a very low memory state of affairs. This behavior may be inconvenient to the user because the oom killer could shut the user's current application once there are a unit different processes within the system, that don't have an effect on the user. Android's low memory driver starts early before a important low memory state of affairs happens and informs processes to avoid wasting their state. If the low memory state of affairs worsens, low memory starts to terminate processes with low importance whose state was saved.

Other changes

In addition to the already delineated kernel changes, there are a unit varied different changes that bit miscellaneous areas of the kernel. Many minor changes area unit delineated during this section.

Binder: in contrast to within the customary UNIX operating system kernel, the IPC mechanism in Android's kernel isn't System V compatible. The kernel facet of the mechanism is predicated on Open Binder, therefore centered on being light-weight weight. So as to reinforce performance, the binder driver uses shared memory to pass the messages between threads and processes.

Logger: Extended kernel work facility with the four work classes: main, system, event and radio. The applying framework uses the system category for its log entries.

Early suspend: Drivers will create use of this capability to try and do necessary work, if a user program desires the system to enter or leave a suspend state.

Alarm: "The alarm interface is comparable to the hrtimer interface however adds support for awaken from suspend. It additionally adds a go on Realtime clock that may be used for periodic timers that require to stay running whereas the system is suspended and not be discontinuous once the wall time is about."

Device & platform support

Android's native platform is ARM, however there exist ports to different platforms like unit and x86 further. The amount of ports grows as automaton becomes a lot of and a lot of fashionable. The list of supported devices grows steady too, as there are a unit a lot of and a lot of automaton powered devices on the market, particularly smartphones with ARM SoCs. A number of those device configurations may be found within the main repository, whereas others area unit created on the market by makers like HTC or Motorola. Porting androidian operating system to a replacement design largely means that porting Dalvik, because the UNIX operating system kernel is perhaps already running on the new platform. The core libraries of Dalvik accept different libraries like OpenSSL and zlib that have to be compelled to be ported within the 1st place. After that, the JNI decision Bridge needs to be ported per the C job conventions of the new design. The last step is porting the interpreter which suggests implementing all Dalvik op codes.

Application Framework APIs

The application framework of androidian operating system provides APIs in varied areas like networking, multimedia, graphical interface, power management and storage access. The libraries within the framework area unit written in Java and run on prime of the core libraries of the androidian operating system Runtime. These core libraries utilize and encapsulate optimized native system libraries like libc, libssl and FreeType. The applying Framework provides managers for various functions like power management, resource handling, system wide notification and window management. Applications area unit purported to use the services of the managers and not use the underlying libraries directly. This fashion it's potential for the managers to enforce application permissions through the means that of the sandbox permission system. The managers will make sure that AN application is allowed to e.g. initiate a telephony or send information over the network.

APIs

Given the huge variety of APIs in androidian operating system, solely many components area unit coated during this section.

API levels

As the development of androidian operating system continues, new apis area unit adscititious and previous apis get marked obsolete and area unit eventually erased. Every androidian operating system version has its own API level and applications will outline a minimum, most and most well-liked API level in their manifest file. The API level changes between major further as minor releases of androidian operating system. **User interface**

The basis of a graphical interface in androidian operating system is that the read category. All visible components of a interface and a few invisible things area unit derived from this category. Androidian operating system provides standardized UI components like buttons, text and video views or a date picker. Read things may be partitioned off to a View Group that permits applying a layout to the views within the cluster. So as to move with the UI, the read category provides hook ways like onClick(), onTouch() and onKey() that area unit referred to as by the underlying framework. UI events may be received by listeners that give the onXYZ() methodology and area unit registered like e.g. the OnKeyListener. Additionally, to the delineated UI components, the framework provides menus and dialogs. Menus will either be choices menus that may be accessed via the menu key, or they're context menus of a read. To tell the user or to get input, AN application will gift a dialog. Androidian operating system provides dialogs for date and time choosing, progress notification and a customizable general dialog with buttons. For user notification exist 3 completely different mechanisms in androidian operating system. The foremost intrusive one is that the notification with a dialog that moves the main target to the dialog, feat the applying within the background.

The toast notification displays a text on prime of this application that enables no interaction and disappears once a given timeout. The standing bar notification leaves this application look unaffected and puts an icon within the standing bar. The toast notification further because the standing bar notification may be used by background services.

Media framework

The media framework supports multiple audio, video and image information formats and includes a media player and an encoder engineered into androidian operating system. The amount of information formats may be extended, however as a minimum the subsequent decodable formats area unit supported:

Image: JPEG (encoder provided), PNG, GIF and BMP

Audio: MP3, OGG Vorbis, MIDI, PCM/WAVE, AAC, AAC+ and AMR (encoder
provided)

Video: H.263 (encoder provided), H.264 and MPEG-4

The MediaPlayer and MediaRecorder categories may be wont to reproduce and record the supported multimedia system information. Additionally to the player and recorder, the android.media package includes specialised categories to play alarms and ring tones or generate image thumbnails and created the camera.

Network

Access to networks and particularly the net is crucial for androidian operating system devices, as several applications depend upon network access. Most automaton devices have multiple technologies on board to realize network access. Smartphones a minimum of have net access via GPRS (2G telephony) or UMTS and frequently LAN is out there too. The ConnectivityManager permits applications to envision the standing of the various access technologies and it informs applications via broadcasted intents concerning property changes. to form use of the network affiliation, automaton provides a broad vary of packages and categories.

java.net.* the quality Java network categories like sockets, easy communications protocol and plain packets android.net.* Extended java.net capabilities

android.net.http.* SSL certificate handling

org.apache.* specialised communications

protocol

android.telephony.* GSM &amp; CDMA specific categories, send text message, signal strength and status info. android.net.wifi.* LAN configuration and standing.

Bluetooth

Android's Bluetooth apis permits applications to look for different devices, combine with them and exchange information. So as to use the Bluetooth capabilities, AN application has to have the BLUETOOTH or BLUETOOTH ADMIN permission in its manifest file granted. The Bluetooth API is found within the android.bluetooth package. The affiliation via a RFCOMM (RS-232 serial line via Bluetooth) compatible BluetoothSocket is initiated by a neighborhood BluetoothAdapter and targeted to a distant BluetoothDevice. Creating the device visible for scans and pairing devices would like user interaction because the permission system asks for granting the required permissions.

Storage & backup

Applications will store and retrieve their information in varied ways that:

Shared Preferences: a category that gives storage for key/value pairs of primitive information sorts. The info may be shared between multiple shoppers within the same method. All information will solely be changed

Internal/external storage: If AN application desires to form positive that no different application or maybe the user will access saved information, it will write this information to the interior storage. Files saved to the interior storage may be created to permit others to access the files. Files that area unit purported to be shared and/or user visible, may be keep to the secondary storage. Files during this {storage area unita|cargo area|cargo deck|cargo hold|hold|enclosure} are continuously in public visible and accessible. Androidian operating system permits the secondary storage to be changed into AN USB mass storage with full access to all or any files during this cargo area.

Database storage: The SQLite information files behind all content suppliers area unit available to applications as AN application personal information backside. Network storage With network access available, androidian operating system applications will get and store their information via sockets, communications protocol connections and different means that from java.net.* and automaton.net.*.

Application information backup: From version 2.2 on androidian operating system provides a mechanism to backup preferences and application information on a distant web site – the cloud. Applications will request a backup and androidian operating system mechanically restores the info on an application set up, if the applying is put in on a replacement device or by request of the applying. The mechanism consists of 3 components on the device: The backup agent, transport and manager.

Applications will announce a backup agent within the manifest file and implement it to produce hooks for the backup manager. If application information changes, the applying will inform the backup manager which can schedule a backup and decision the according backup agent hook. The backup mechanism doesn't permit on demand scan and write access to the insured information as it’s continuously the backup manager’s call once to perform a restore or a backup. The backup transport is accountable to transfer the info from or to the remote web site. There’s no guarantee that the backup capability is out there on a tool because the backup transport depends on the device manufacturer and therefore the service supplier. What is more there's no security assurance for the info within the backup.

Other APIs

Supplementary to the already mentioned apis, androidian operating system provides a large vary of different apis for several completely different areas. A number of those packages area unit listed below:

Location &amp; Maps Applications: will use the LocationManager and categories from android.location to get the device’s location directly on demand or by a broadcast. The external com.google.android.maps package provides mapping facilities for
applications.

Search: The system wide search on androidian operating system devices includes not solely contacts, internet search and such, however may be extended by applications to form their content supplier information searchable. The SearchManager provides unified dialogs and additional practicality like voice rummage around for all applications.

WebKit: For browsing sites, androidian operating system provides among different things a WebKit based mostly markup language renderer, a JavaScript engine (V8) and a cookie manager. These facilities may be utilized by applications to produce browser practicality within the applying.

Speech: The androidian operating system.speech package puts applications within the position to form use of server facet speech recognition services. A text-to-speech API provides the means that to show text into audio files or reproduce the result directly.

C2DM: Cloud to Device electronic messaging – A new and in android two.2 fairly restricted capability to send information like URLs or maybe intents to the device.

2.2. Android Open Accessory [1][2][3][4]

Android Open Accessory give us the power to interact with android powered device to external USB hardware in special mode. Android phone when in accessory mode the connected accessory work as USB accessory role. When android USB accessory is connected to an android powered phone then it will search the accessory mode in the android phone with the help of android open accessory protocol. For charging power, accessory should provide the 500ma at 5V.The before android device are unavailable to make connection to the USB connected device. This problem has been overcomes with the release of Android open accessory. An API level 12 & higher can use the android open accessory protocol.

2.3. Android Bluetooth Libraries [1][2][3][4]

Package containing android libraries is in – android.bluetooth package

Get Adapter -> Get Paired Devices -> Search Bluetooth Android connects arduino via the android bluetooth package btInterface = BluetoothAdapter.getDeafaultAdapter(); Obtain a list of paired devices with a reference to the adapter.

pairedDevices = btInterface.getBondedDevices();

Search our firefly bluetooth from a list of devices

IIterator<BluetoothDevice> btlist = pairedDevices.iterator();

While(it.hasNext()){

BluetoothDevice bd = it.next();

If(bd.getName().equalIgnoreCase(BluetoothName))

Initiate Connection -> Create Socket -> Connect Socket After bluetooth is found, initiate connection with the function connectToBluetooth(bd); When connecting to the Bluetooth createRFComSocketToServiceRecord()method is used socket = bd.cresteRfcommSocketToServiceRecord

(UUID.fromString("00001101-0000-1000-8000-00805F9B34FB"));

Connect to the socket by using the connect() method.

Socket.connect():

Monitoring the bluetooth connection

To do this, register two bluetooth events:

ACTION_ACL_CONNECTED and ACTION_ACL_DISCONNECTED

When the device are connected, the handleConnected() method is invoked.

"android.bluetooth.device.action.ACL_Connected"))

{ handleConnected();

When the remove device disconnects, the handleDisconnected() method is invoked.

"android.bluetooth.device.action.ACL_DISCONNECTED"))

{ handleDisconnected();

Once handleConnected() is invoked, the connection is established.

Set up the input and output streams for communication between android and android.

Is = socket.getInputStream();

Os = socket.getOutputStream();

Error Event

Call close method to disconnect

Socket.close();

Bluetooth permission

Defined Permission in AndroidManifest.xml file:

<uses-permission android:name=
“android.permission.BLUETOOTH”>

</usepermission>

2.4. Arduino [8]

The arduino was born at the Interaction Design Institute in the city of Ivrea, Italy in 2005. Prof. Massimo Banzi of Interaction Design Institute was looking for a solution for an expensive microprocessor.

In his mind he wanted to make something that is cheap and easy to use and can be able to integrate with the other hardware. So with the help of David Cuartielles, an engineer visiting the Interaction Design Institute from the Malono University, Sweden and with the help of two computer science students they came up with a simple and efficient microcontrollers and they decided to name it after a local bar named arduino.

As time passes they decided to make arduino an open source hardware as the Interaction Design Institute is closing down. They feared that their valuable design can be go wasted, so they made it public for all the world by open sourcing it.

With some fee anybody can produce that design microprocessor and because to its simple and easy way to install and integrate with others made it popular in the open source world. An arduino microprocessor is of the ATmega microprocessor family.

3. Software and Hardware Specification

Software used

S/W Tool: Android Studio

Operating System: Windows 7 (x86 & x64)

S/W Tool: Arduino IDE

Hardware requirements

Android Operating System (2.3.3 – 4.4.4)

1.0GHz or faster processor

256 MB RAM minimum

Minimum 500MB of available hard disk space

Arduino Uno R3 microprocessor

Arduino Uno shield

L298 Motor shield

Stepper motor

Servo motor

Bluetooth module- class II

Ultrasonic sensor

5-8V External Power Source/Battery

4. Project Design and Implementation [5][6][7][9]

For this project we are following the agile model. We created modules individually and performed manual testing, since most of the modules were of different platforms.

4.1. Approach to Design

Enlisted our design approaches for the application. These were chosen after thorough discussion.

4.1.1 Scrum Model

A scrummage project involves a cooperative effort to form a replacement product, service or alternative result as outlined within the project vision statement. Project are compact by constraints of your time, cost, scope, quality, resources, structure capabilities and alternative limitations that build them troublesome to arrange execute manage and ultimately succeed. However, successful implementation of the results of a finished project provides important business edges to a company. It's so necessary for organizations to pick Associate in Scrum apply an acceptable project management methodology.

Scrum is one among the foremost in style methodologies. It's Associate in Scrum is adaptable, fast, versatile and effective methodology designed to deliver important price quickly and throughout a project. Scrummage ensures transparency in communication
Associate in Scrum creates a setting of collective respondents and continuous progress.

The scrummage framework as outlined within the SBOK guide is structured in such the way that it supports product and repair development all told the kinds if industries and in any style of project, no matter its quality

A key strength of scrummage lies in its use of cross-functional, self-organized and sceptered groups , their add to short, focused work cycles known as Sprints figure below offer an outline of a scrummage project's flow

The scrummage cycle begins with a stake holder meeting, throughout that the project vision is formed. The merchandise owner then develops a prioritized product backlog that contains a prioritized list of business and project necessities written within the kind of user stories.

Each Sprint begins with a Sprint designing conferences throughout that high priority user stories are thought-about for inclusion. A sprint typically lasts between one and 6 weeks and involves the scrummage team operating to form probably shippable deliverables or product increments. Throughout the sprint, short extremely targeted daily standup conferences are conducted wherever team members discuss daily progress.

Towards the top of the sprint, a sprint review meeting is command throughout that the merchandise owner and relevant stakeholders are provided an illustration of the deliverables. The merchandise owner accepts the deliverables providing they meet the predefined acceptance criteria. The sprint cycle ends with a retrospect sprint meeting wherever the team discusses ways in which to boost processes and performance as they move forward into the next sprint.

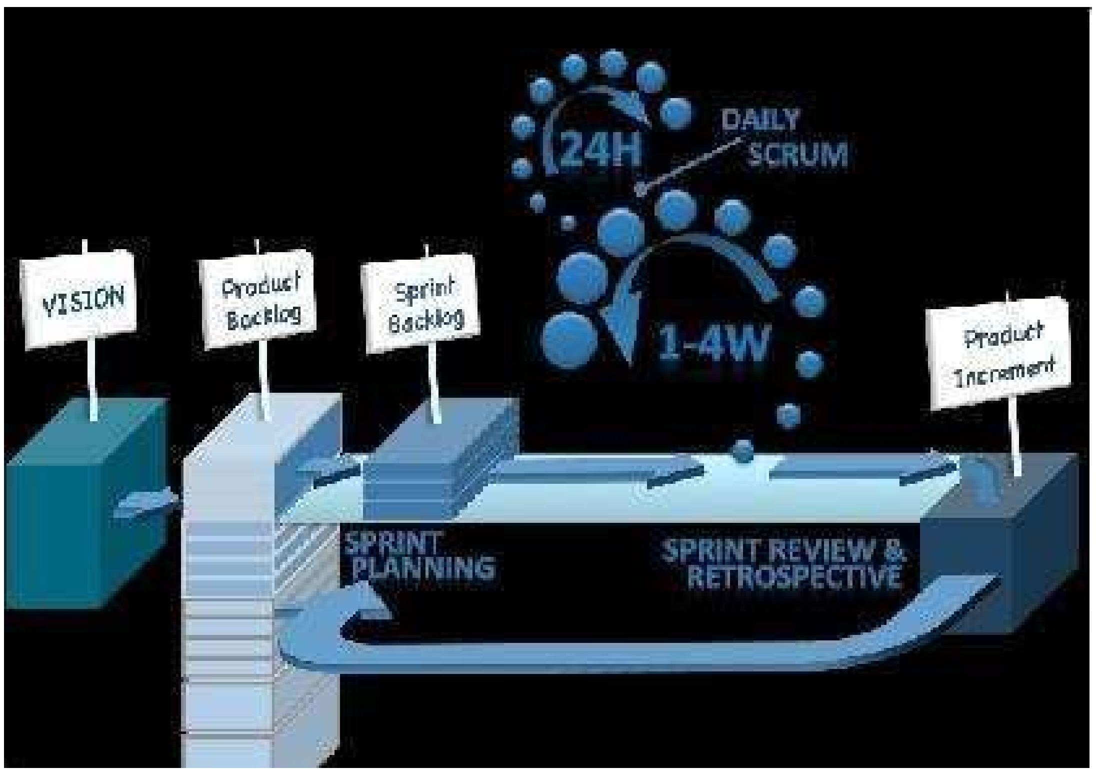


Fig.4.1- Scrum model phases

Scrum phases

Initiate

Plan and estimate

Implement

Review and retrospect

5. Experimentation

5.1. Android Studio

Android Studio is that the official IDE for android application development, supported IntelliJ idea. On prime of the capabilities you expect from IntelliJ, android Studio offers:

- Flexible Gradle-based build system
- Build variants and multiple apk file generation
- Code templates to assist you build common app options

- Rich layout editor with support for drag and drop theme piece of writing lint tools to catch performance, usability, version compatibility, and alternative issues.
- ProGuard and app-signing capabilities
- Built-in support for Google Cloud Platform, creating it straightforward to

integrate Google Cloud electronic communication and

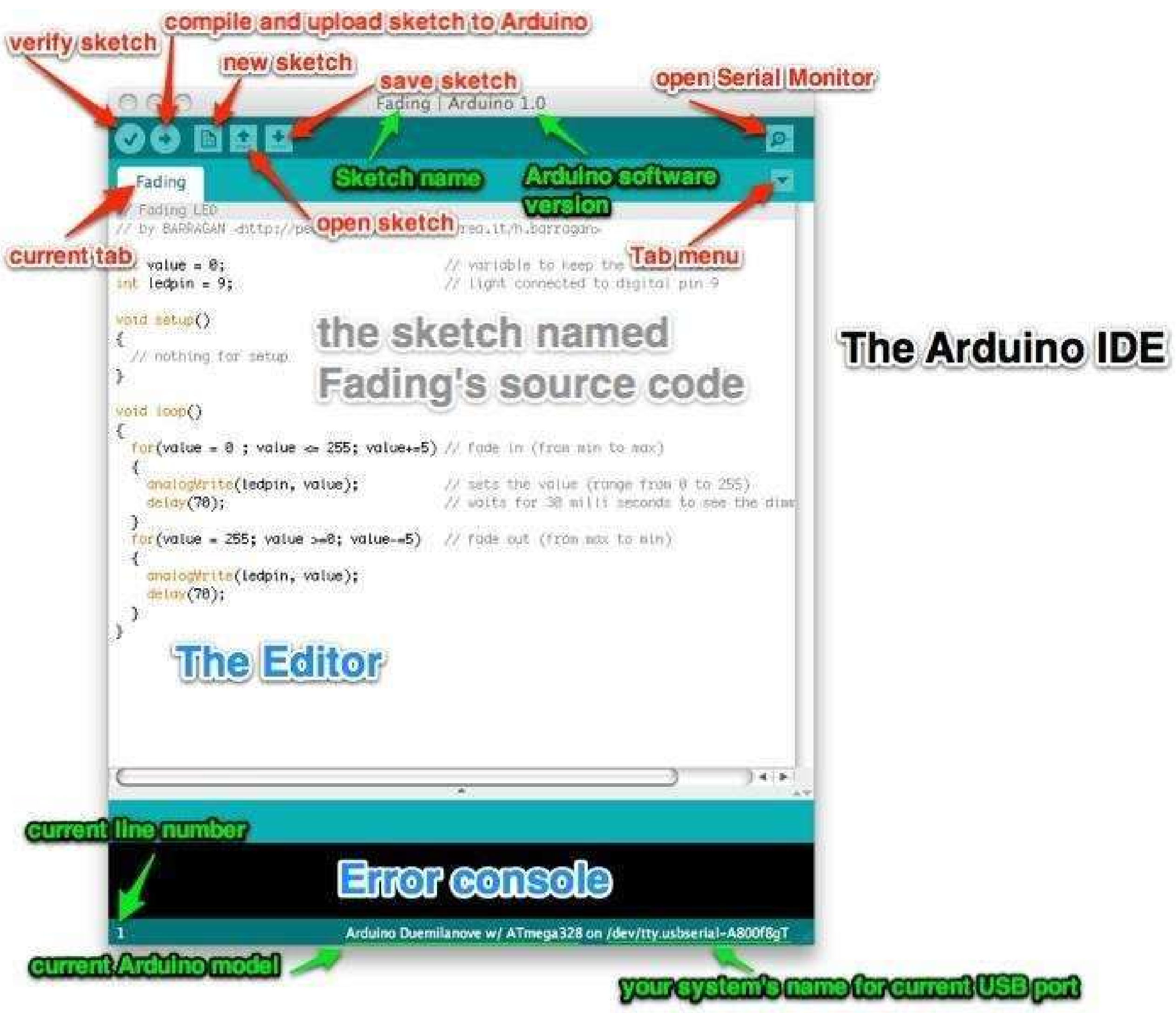
App Engine And much more.

5.2. Arduino IDE

The Arduino integrated development environment (IDE) could be a cross-platform application written in Java, and derives from the IDE for the process programing language and also the Wiring projects. It’s designed to introduce programming to artists and different newcomers unfamiliar with code development. It includes a code editor with options like syntax highlighting, brace matching, and automatic indentation, and is additionally capable of compilation and uploading programs to the board with one click. A program or code written for Arduino is named a "sketch".

Arduino programs are written in C or C++. The Arduino IDE comes with a code library referred to as "Wiring" from the first Wiring project that makes several common input/output operations a lot of easier. The users would like solely to outline 2 functions to create a possible cyclic government program:

setup (): a function run once at the beginning of a program that may initialize settings loop (): a function referred to as repeatedly till the board powers off+ programming languages.



```
// Fading LED
// by BARRAGAN <http://pe                               rea.it/h.barragan>

  value = 0;                               // variable to keep the
int ledpin = 9;                            // light connected to digital pin 9

void setup()
{
  // nothing for setup
}

void loop()
{
  for(value = 0 ; value <= 255; value+=5) // fade in (from min to max)
  {
    analogWrite(ledpin, value);           // sets the value (range from 0 to 255)
    delay(70);                            // waits for 30 milli seconds to see the dimm
  }
  for(value = 255; value >=0; value-=5)   // fade out (from max to min)
  {
    analogWrite(ledpin, value);
    delay(70);
  }
}
```

```
Arduino Duemilanove w/ ATmega328 on /dev/tty.usbserial-A800f8gT
```

Fig.5.1- Arduino IDE

5.3. Arduino Uno R3 microprocessor

Uno means one in Italian, as uno is the first microcontroller of arduino family.

Arduino uno r3 microcontroller board work on ATmega-328

ATmega 328 microcontroller

5V operating voltage

7-12V input voltage

6-20V input voltage (limits)

14(6 PWN outputs) Digital Pins

6 Analog input pins

40mA DC current per I/O pin

50mA DC current for 3.3V pin

32KB (0.5 used for bootloader) flash memory

2KB SRAM

1KB EEPROM

16 MHz clock Speed

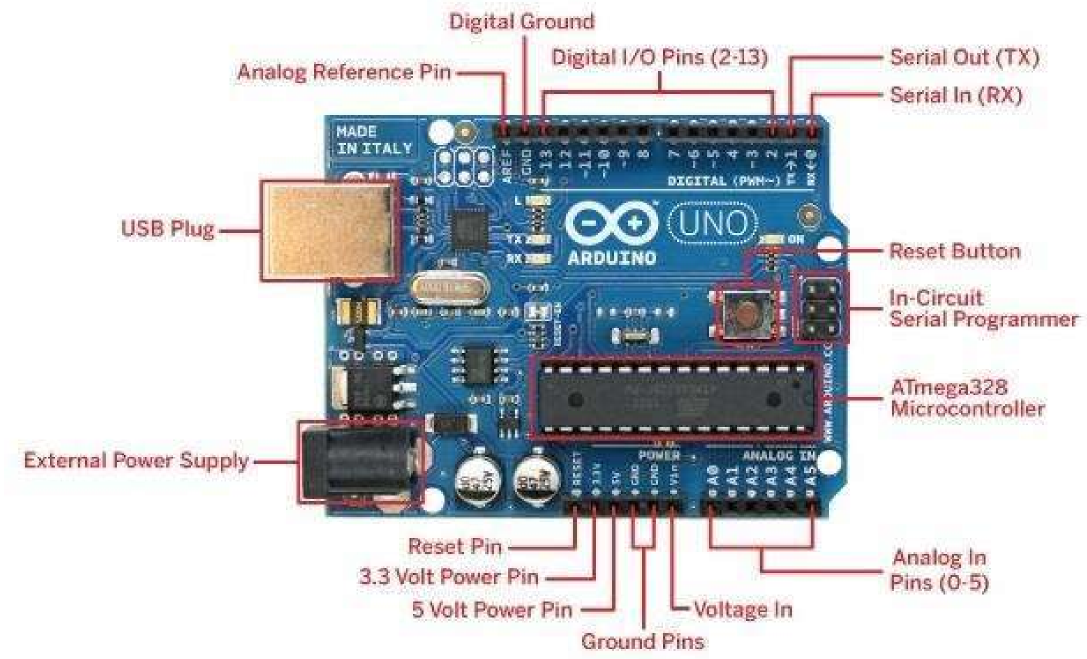


Fig 5.2- Arduino Microprocessor

5.4. L298 Motor Shield

L298 motor shield is a dual full bridge driver designed to drive inductive load i.e. stepper motor, relays & solenoids. It gives you the control to drive two DC motor with arduino microcontroller and provide you the royalty to control the direction and speed of the motor.

5V (from arduino) Logic control voltage

4-8V ~ 35V (from external power source) motor driven voltage

36mA >= Logic supply current Iss

2A >= motor driver current Io

25W Maximum power consumption (T=75C)

PWN, PLL Speed control mode

High 2.3V=<Vin=<5V

Low -0.3V=<Vin=<1.5V

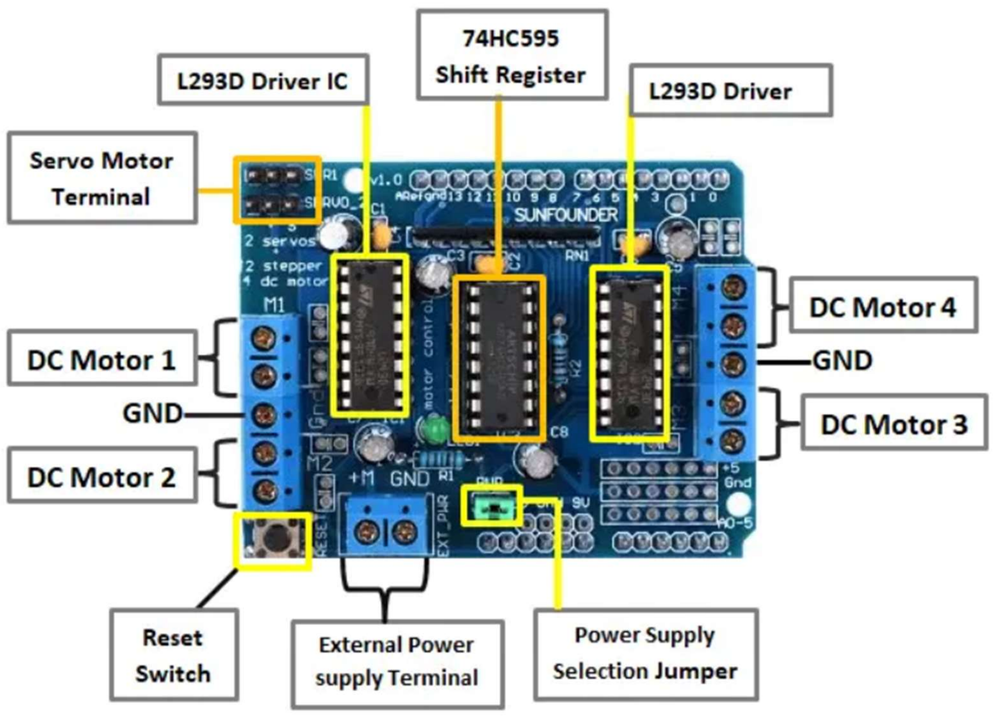


Fig 5.3- Motor shield

5.5. Stepper Motor

Stepper motor works as an electrometrical device that with the help of electric pulses. Workout its mechanical operations the stepper motor rotates only when proper/clear. Sequence signal logic applied through the motor shield**.**

The sequence of electric pulses is directly connected to the direction of the stepper motor. The speed of the stepper depends upon the frequency of input pulses and length of rotation directly depends upon the applied no. of input pulses.

Stepper motor is of good choice when we want to control the rotation angle, speed, position and synchronism.

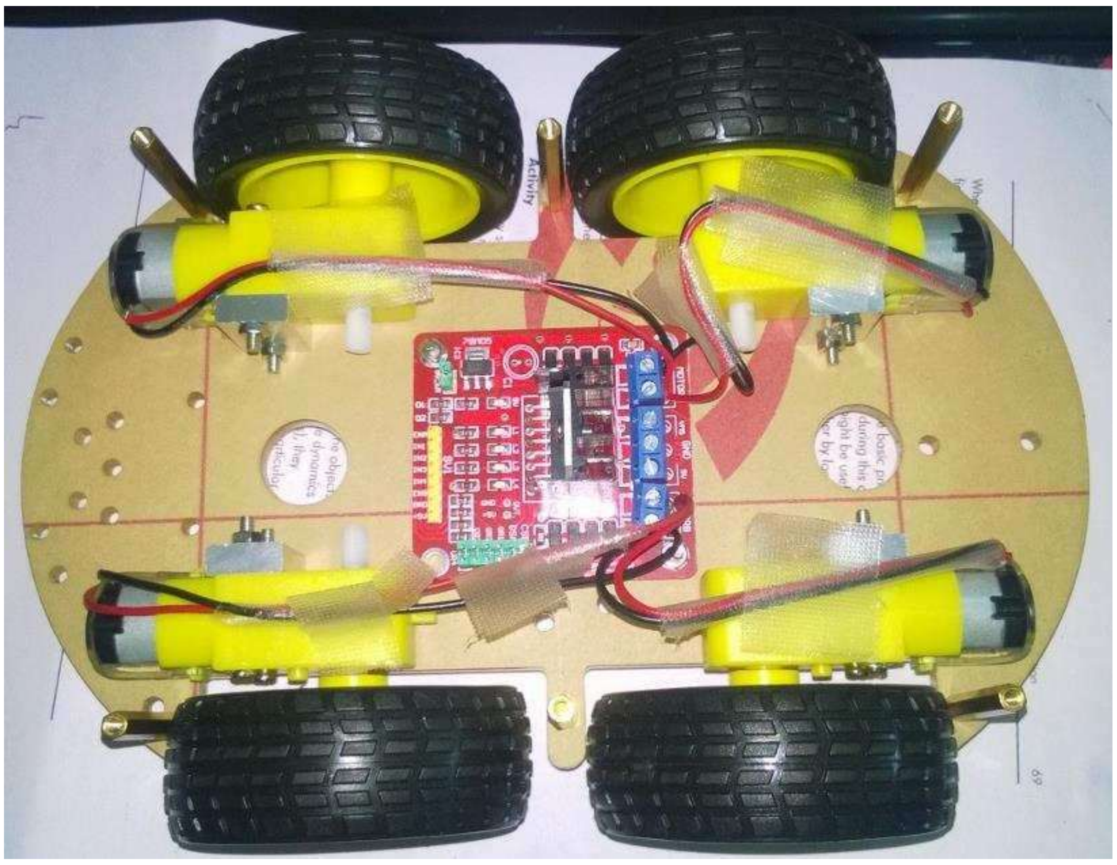


Fig 5.4- Stepper motor connected with motor shield

5.6 Servo motor

Servo motor (3 wire dc motor) is used for a low power (~100W) control application balanced servomotor is ideally suited as it can be driven by means of a relatively rugged (drift free) ac amplifier. The motor torque can be easily controlled by varying the magnitude of the ac voltage applied to the control phase of the motor.

A three wire dc servo motor has

DC motor

A geartrain

Limit stops (after which shaft can't turn)

A potentiometer (position feedback)

Three wires of servo motor are described as one for power, one for ground, one for control input which determines the position motor should servo.

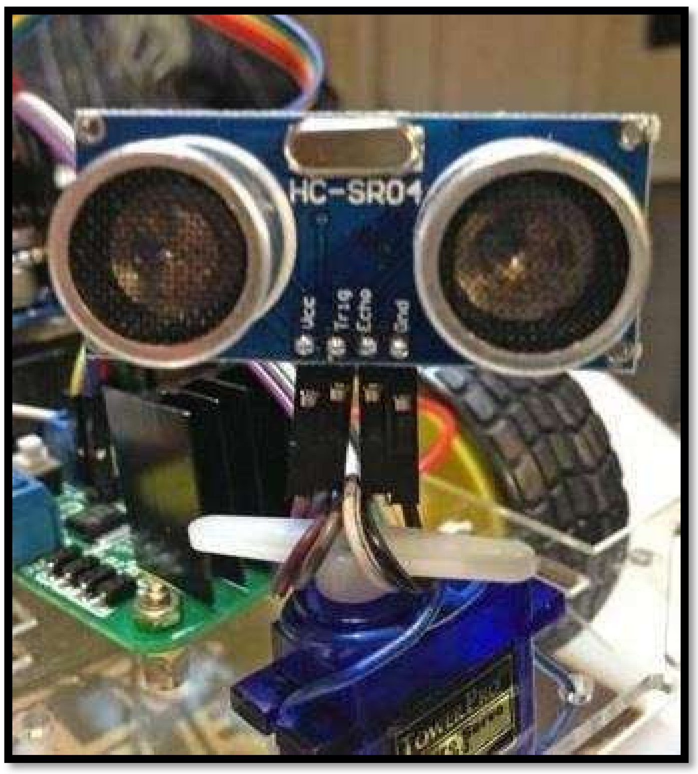


Fig 5.5 Servo motor with ultrasonic sensor

5.7. Screenshots of Application

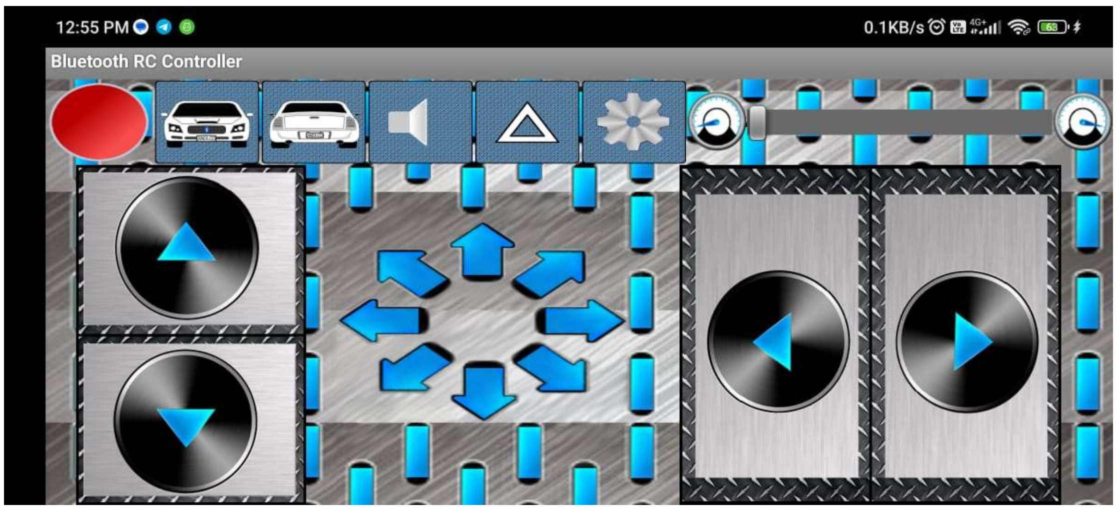


Fig 5.6. Android controller Splash Screen

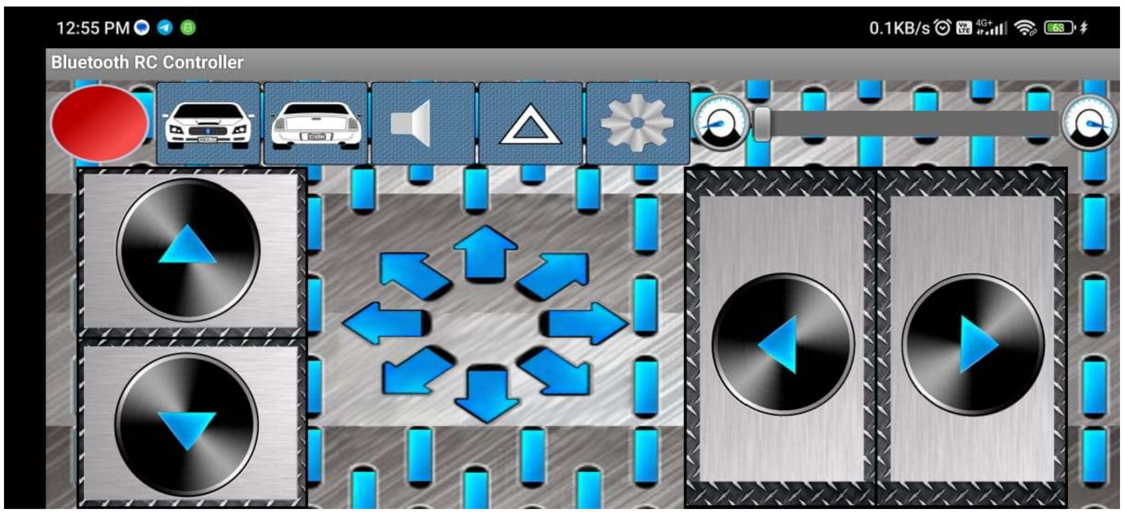
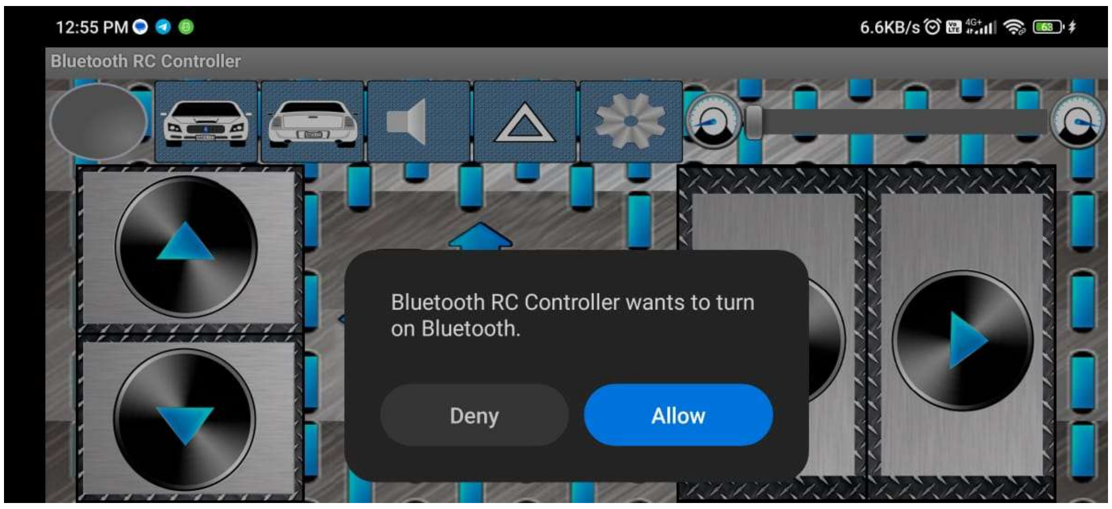
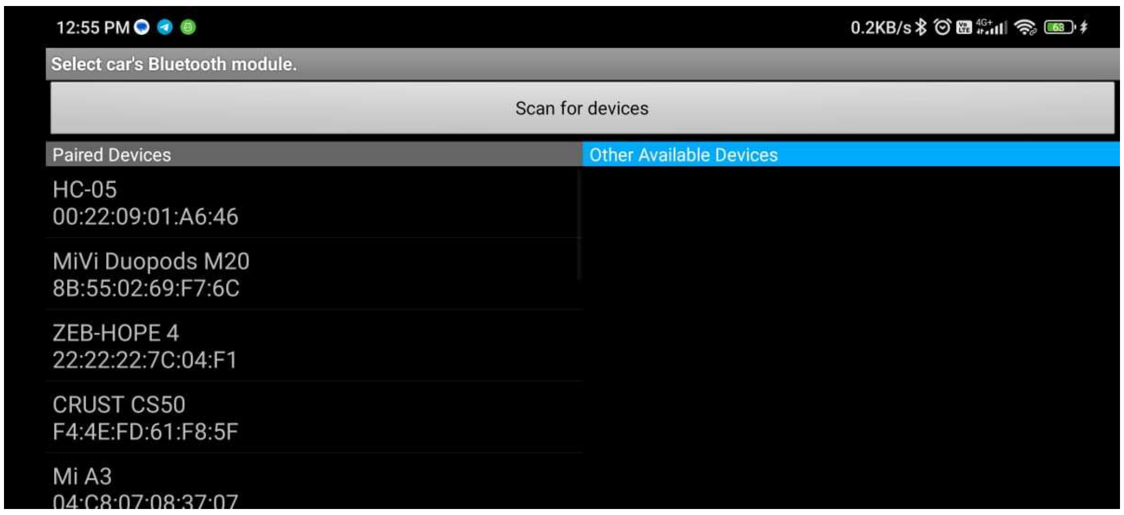


Fig 5.7. Android controller main interface



Bluetooth RC Controller wants to turn on Bluetooth.

Fig 5.8. Android controller initiating the bluetooth connection



HC-05
00:22:09:01:A6:46

MiVi Duopods M20
8B:55:02:69:F7:6C

ZEB-HOPE 4
22:22:22:7C:04:F1

CRUST CS50
F4:4E:FD:61:F8:5F

Mi A3
04:C8:07:08:37:07

Fig 5.9. Scanning for the arduino car’s bluetooth

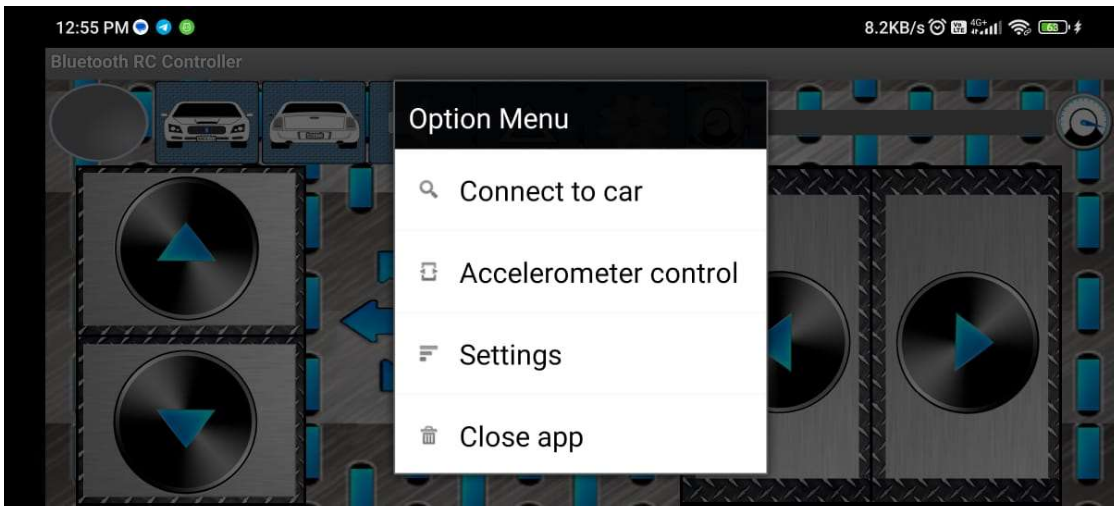


Fig 5.10. Android controller trying to connect with arduino car

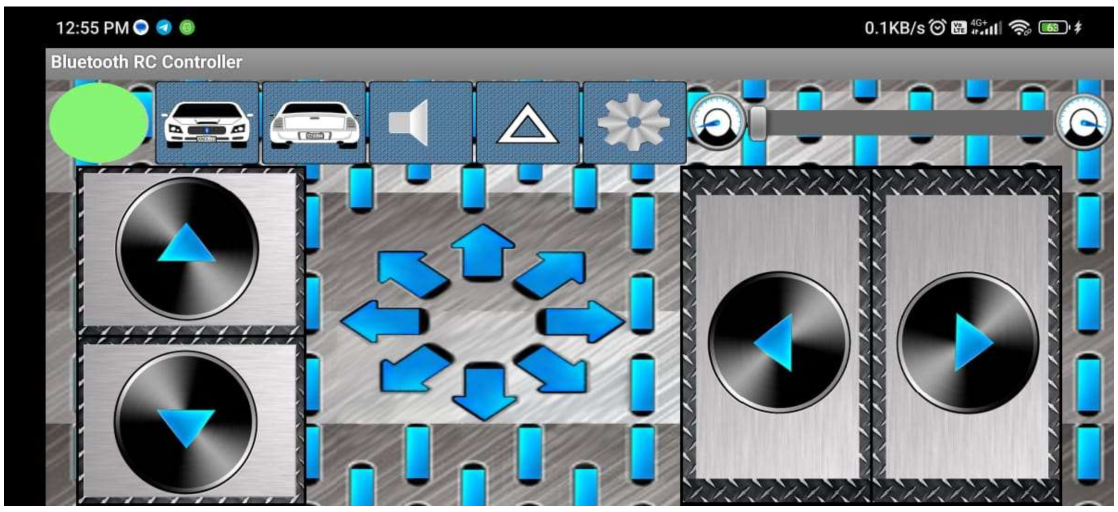


Fig 5.11. Android controller disconnected with arduino car

6. Test cases

Manual testing was chosen as the method for testing. The randomness of the test cases is one of the characteristic of manual testing.

Ref No.	Test Data	Expected Outcome	Final Result
1.	Install the Android Application on the Android Operating System (2.3.3-5.1)	Application is installed on the Android Device Successfully	Pass
2.	Switch ON the external power supply/battery key of the Arduino car	Power supply has been reached to every microprocessor of Arduino car	Pass
3.	Connect the bluetooth module to the arduino shield	Light of red colour starts to blink from the bluetooth module	Pass
4.	Start the android application by clicking the icon	Application is started on the android OS	Pass
5.	Splash Screen Of the Application	Splash Screen is coming after the starting of Application	Pass
6.	Click on the search new bluetooth device	Bluetooth device list has been generated	Pass
7.	Click the connection button on the application interface	A message of Connected will show onto the application interface/Blinking	Pass

		stops of bluetooth module	
8.	Click on to the forward button of the android application interface	Arduino car starts to move forward & going forward message will generate in to the message box	Pass
9.	Click on to the Stop button of the android application interface	Arduino car stop moving & stopping message will generate in to the message box	Pass
10.	Click on to the back button of the android application interface	Arduino car starts to move forward & going backwards message will generate in to the message box	Pass
11.	Click on to the Stop button of the android application interface	Arduino car stop moving & stopping message will generate in to the message box	Pass
12.	Click on to the Right button of the android application interface	Arduino car starts to move in right direction & turning right message will generate in to the message box	Pass
13.	Click on to the Stop button of the android application interface	Arduino car stop moving & stopping	Pass

		message will generate in to the message box	
14.	Click on to the left button of the	Arduino car starts to move in left direction & turning left message	Pass
	android application interface	will generate in to the message box	
15.	Click on to the Stop button of the android application interface	Arduino car stop moving & stopping message will generate in to the message box	Pass
16.	Take the android phone beyond 10 feet & give above used command of direction/Button	Unable to connect to the arduino device	Pass
17.	Click on to the Disconnect Button	Bluetooth module start to blink/closing connection & then disconnected message will generate in to the message box of android box	Pass
18.	Close the application on to the android device	Bluetooth module still blinking	Pass

19.	Switch OFF the Arduino car	All the blinking light of microprocessor immediately faded	Pass

7. Discussion of Results

In the course of developing this project we have achieved the following milestones.

The android guided arduino car has been created successfully and testing has been done for all the known cases regarding the usage of project.

We has successfully created an android application that will provide user an interface to interact with the arduino powered car. The interface is easy to use and provide feedback from the arduino microprocessor through the bluetooth after giving instruction to arduino for various actions through interface via bluetooth module.

An appropriate program in the arduino microprocessor to interact with the android controller has been created successfully. The program has been successfully complied through arduino IDE to the arduino microprocessor & loaded in to it after proper checking of logic to decrease any loss/damage of hardware.

We have been able to successfully implement the ultrasonic sensor with servo motor in arduino car to save the car from collision.

The project has completed its aim to designing an android interface, arduino bot and write program in to the arduino microprocessor. Arduino car contains arduino microcontroller with basic mobility features. Arduino programs contains instructions mediating between android controller and Arduino car. Android mobile controller uses different mobile sensors to supervise motion.

8. Conclusion

The project titled Android guided arduino car is an application based on popular open source technologies- android & arduino. The aim of the project was to create an arduino integrated car that has to be controlled through an application that runs on the android operating system. Since most of the application in the android market were unable to provide such simplicity that we have covered in this project.

The project has been completed with success with the utmost satisfaction. The constraints square measure met and overcome with success. The system styled /is meant/ is intended as find it irresistible was set within the design section. The project offers smart plan on developing a full-fledged application satisfying the user needs.

The system is extremely versatile and versatile. This code encompasses an easy screen that permits the user to use with none inconvenience. Validation checks iatrogenic have greatly reduced errors. Provisions are created to upgrade the code. The applying has been tested with live information and has provided a prosperous result. Thence the code has proven to figure expeditiously.

The system created met its objectives, by being straightforward to use, implement and secure. This code is developed with measurability in mind. Further modules may be simply other once necessary. The code is developed with standard approach. All modules within the system are tested with valid information and invalid information and everything work with success.

However there\'s still lots of scope for future improvement and add ons in practicality. A number of the foremost ones being mobile application development for different mobile software package

9. Future Prospects

The Future Prospects are as follows:

Mobile application of this project can be brought on other mobile operating system like iOS & windows phone.

Using of Wi-Fi direct/cellular network instead of bluetooth module, so that the range of the arduino car be increased to larger scale.

Application the project concept of guiding through mobile device to quadcopter, so that we can cover the aerial view of the surrounding.

With the help of DIP we can install a camera on the car which will provide us the view of camera on the mobile device.

Due to time & resource constraints these ideas where not brought to light in the project.

10. References

[1] Getting Started with Android referred from developer.android.com/.

[2] J. F. DiMarzio, "Android - A Programmer's Guide 1st Edition", 2010, Tata McGraw - Hill Education ISBN 9780071070591.

[3] Ed Burnette, "Hello, Android: Introducing Google's Mobile Development Platform 3rd Edition", 2011, Pragamatic publications.

[4] Reto Meier, "Professional Android 4 Application Development", 2012, Wiley India.

[5] Referred the book written by Roger Pressman, titled Software Engineering - a practitioner's approach.

[6] Gerhard Fischer, "The Software Technology of the 21st Century: From Software Reuse to Collaborative Software Design"

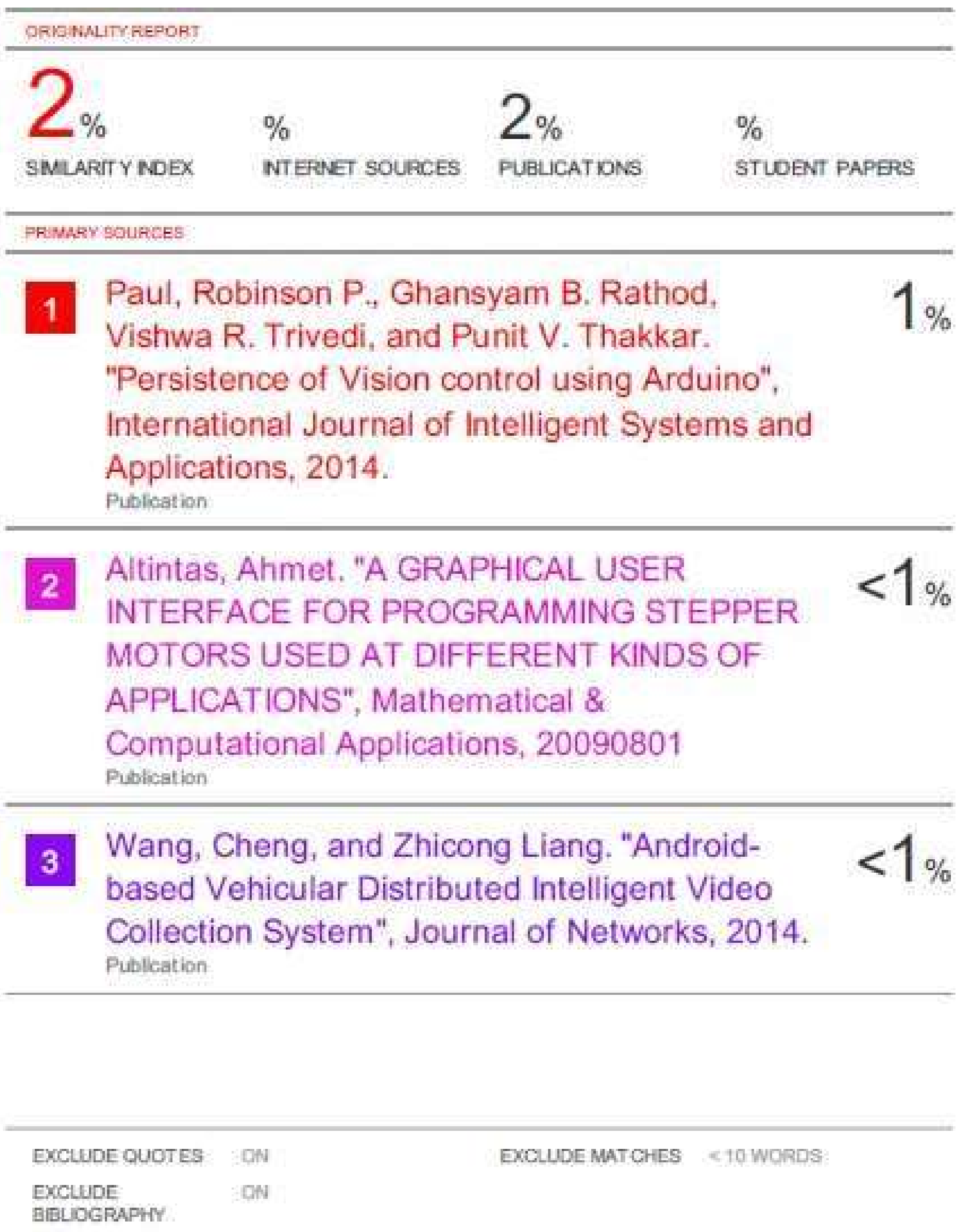
[7] K.K.Aggarwal & Yogesh Singh, "Software Engineering", 2013, New Age

International Publication.

[8] Getting Started with Arduino referred from http://www.arduino.cc//.

[9] www.scrumstudy.com/SCRUMstudy, "A Guide to the Scrum Body of Knowledge (SBOK™ GUIDE), 2013, VMEdu, Inc.

11. Plagiarism Report



ORIGINALITY REPORT

2%	%	2%	%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Paul, Robinson P., Ghansyam B. Rathod, Vishwa R. Trivedi, and Punit V. Thakkar. "Persistence of Vision control using Arduino", International Journal of Intelligent Systems and Applications, 2014. Publication	1%
2	Altintas, Ahmet. "A GRAPHICAL USER INTERFACE FOR PROGRAMMING STEPPER MOTORS USED AT DIFFERENT KINDS OF APPLICATIONS", Mathematical & Computational Applications, 20090801 Publication	<1%
3	Wang, Cheng, and Zhicong Liang. "Android-based Vehicular Distributed Intelligent Video Collection System", Journal of Networks, 2014. Publication	<1%

EXCLUDE QUOTES	ON	EXCLUDE MATCHES	< 10 WORDS
EXCLUDE BIBLIOGRAPHY	ON		