**AUTOMATED AGRICULTURE ASSISTANT ROVER**

**A PROJECT REPORT**

***Submitted by***

**AVINASH.R (111517107009)**

**DINESH KUMAR.T (111517107012)**

**DINESH.S (111517107013)**

**GIRIDHARAN.KGK (111517107016)**

***In partial fulfillment for the award of the degree***

***of***

**BACHELOR OF ENGINEERING**

**IN**

**ELECTRONICS AND INSTRUMENTATION ENGINEERING**

**R.M.D.ENGINEERING COLLEGE**

**KAVARAIPETTAI**

**ANNA UNIVERSITY: CHENNAI 600 025**

**MARCH 2021**

|  |  |
| --- | --- |
| **ANNA UNIVERSITY – CHENNAI-600 025**  **BONAFIDE CERTIFICATE**  Certified that this project report **“AUTOMATED AGRICULTURE ASSISTANT ROVER”** is the bonafide work of **AVINASH.R (111517107009),DINESH KUMAR.T (111517107012), DINESH.S (111517107013), GIRIDHARAN.KGK (111517107016)**who carried out the project work under my supervision. | |
| **SIGNATURE OF THE HOD**  Dr. A. CHILAMBUCHELVAN M.E, Ph.D.,  PROFESSOR AND HEAD  DEPARTMENT OF ELECTRONICS AND INSTRUMENTATION ENGINEERING  R.M.D. ENGINEERING COLLEGE,  R.S.M. NAGAR,  KAVARAIPETTAI - 601206. | **SIGNATURE OF THE SUPERVISOR**  Mrs. PADMAVATHI.N,  ASSISTANT PROFESSOR  DEPARTMENT OF ELECTRONICS AND INSTRUMENTATION ENGINEERING  R.M.D. ENGINEERING COLLEGE,  R.S.M. NAGAR,  KAVARAIPETTAI – 601206. |

**CERTIFICATE OF EVALUATION**

**College Name** R.M.D.ENGINEERING COLLEGE

**Department** ELECTRONICS & INSTRUMENTATION ENGINEERING

**Semester** 08

|  |  |  |
| --- | --- | --- |
| **Title of Project** | **Name of the Students with Register Numbers** | **Name of the Supervisor with Designation** |
| Automated Agriculture Assistant Rover | AVINASH.R  (111517107009)  DINESH KUMAR.T  (111517107012)  DINESH.S  (111517107013)  GIRIDHARAN.KGK  (111517107016) | Mrs. PADMAVATHI.N,  Associate Professor,  Dept.of Electronics& Instrumentation Engineering |

The report of the project work submitted by the above students in partial fulfillment for the award of **Bachelor of Engineering Degree in ELECTRONICS & INSTRUMENTATION ENGINEERING** of Anna University was evaluated and confirmed to be the report of the done by the above students and then evaluated.

**Submitted the project during the Viva-Voce held on**

**INTERNAL EXAMINER EXTERNAL EXAMINER**

**ACKNOWLEDGEMENT**

The success and final outcome of this project required a lot of guidance, Support and kind cooperation from many, for successful completion. We wish to express our sincere thanks to all those who were involved in the completion of this project.

It is our immense pleasure to express our deep sense of gratitude to our chairman **Thiru R. S. Munirathinam**, our vice chairman **Thiru R. M. Kishore**, and our director **Thiru R. Jothi Naidu** for the facilities and support given by them in the college.

We are extremely thankful to our principal **Dr. N. Anbuchezhian, M.S, M.B.A, M.E, Ph.D.**, for giving us an opportunity to serve the purpose of education.

We are indebted to **Dr. A. Chilambuchelvan, M.E, Ph.D.**, Professor and Head of the Department of Electronics & Instrumentation Engineering, for his valuable guidance and useful suggestions during the course of the project.

We are thankful to our project coordinator **Dr. N. Vini Antony Grace, B.E, M.E**, (**Ph.D.)** , Assistant Professor, in the Department of Electronics & Instrumentation Engineering, for providing the necessary guidance and constant encouragement for successful completion of this project on time.

We extend our sincere thanks and gratitude to our project guide **Mrs. N.PADMAVATHI**, **M.E (Ph.D)** Associate professor in the Department of Electronics & Instrumentation Engineering, who took keen interest on our project work and guided us all along, by providing necessary information till the completion of our project work.

Last but not the least, I wish to thank all the teaching and non-teaching staff of Electronics & Instrumentation Department, for their blessings and for their constant support throughout our dissertation.

|  |  |  |
| --- | --- | --- |
| **TABLE OF CONTENTS** | | |
| **CHAPTER** | **TITLE** | **PAGE NO** |
| CHAPTER 1 | INTRODUCTION |  |
| 1.1 | INDUSTRIAL AUTOMATION | 01 |
| 1.2 | IOT | 03 |
| 1.3 | ROVER ROBOTICS | 04 |
| 1.4 | CONVENTIONAL AGRICULTURE METHODS | 04 |
| 1.5 | AUTOMATION IN AGRICULTURE | 06 |
| CHAPTER 2 | INTERPRETATION SURVEY AND ANALYSIS |  |
| 2.1 | INDUSTRIAL AUTOMATION GRAPH | 07 |
| 2.2 | IOT GRAPH | 08 |
| 2.3 | ROVER ROBOTICS GRAPH | 08 |
| 2.4 | AUTOMATION IN AGRICULTURE GRAPH | 09 |
| CHAPTER 3 | BLOCK DIAGRAM |  |
| 3.1 | DESCRIPTION | 11 |
| 3.2 | RESULTS AND DISCUSSION | 12 |
| CHAPTER 4 | LITERATURE SURVEY |  |
| 4.1 | EXISTING SYSTEMS | 15 |
| 4.2 | ADVANTEGES OVER EXISTING SYSTEMS | 19 |
| CHAPTER 5 | AUTOMATED ROVER |  |
| 5.1 | PROPOSED SYSTEM | 20 |
| 5.2 | REQUIREMENTS | 20 |
| CHAPTER 6 | HARDWARE DESCRIPTION |  |
| 6.1 | ARDUINO UNO | 21 |
| 6.2 | ESP32 | 22 |
| 6.3 | L293D MOTOR SHIELD FOR ARDUINO | 22 |
| 6.4 | REES52 SOIL MOISTURE SENSOR | 23 |
| 6.5 | NRF24L01+ RADIO TRANS RECEIVER MODULE | 23 |
| 6.6 | BUZZER AND MIC MODULE | 24 |
| 6.7 | TCS230 COLOUR SENSOR | 25 |
| 6.8 | 9v DC MOTOR | 26 |
| 6.9 | 12v LEAD ACID BATTERY | 26 |
| CHAPTER 7 | SOFTWARE DESCRIPTION |  |
| 7.1 | ARDUINO IDE | 27 |
| 7.2 | ANDROID STUDIO | 42 |
| CHAPTER 8 | REFERENCES | 55 |

**Table of Figures**

|  |  |  |
| --- | --- | --- |
| **S.No.** | **Contents** | **Page Number** |
| 1.0 | INDUSTRIAL AUTOMATION | 02 |
| 1.1 | IOT | 03 |
| 1.2 | ROVER ROBOTICS | 04 |
| 1.3 | CONVENTIONAL AGRICULTURE | 05 |
| 1.4 | AUTOMATION IN AGRICULTURE | 06 |
| 1.5 | INDUSTRIAL AUTOMATION GRAPH | 07 |
| 1.6 | IOT GRAPH | 08 |
| 1.7 | ROVER ROBOTICS GRAPH | 09 |
| 1.8 | AUTOMATION IN AGRICULTURE GRAPH | 10 |
| 1.9 | BLOCK DIAGRAM | 11 |
| 2.0 | ARDUINO UNO | 21 |
| 2.1 | ESP32 | 22 |
| 2.2 | L293D MOTOR DRIVE | 22 |
| 2.3 | SOIL MOISTURE SENSOR | 23 |
| 2.4 | NRF24L01+RADIO TRANS RECEIVER MODULE | 24 |
| 2.5 | BUZZER AND MIC MODULE | 25 |
| 2.6 | TCS230 COLOUR SENSOR | 25 |
| 2.7 | 9v DC MOTOR | 26 |
| 2.8 | 12v LEAD ACID BATTERY | 27 |
| 2.9 | ARDUINO IDE | 42 |
| 3.0 | THE PROJECT FILES IN ANDROID VIEW | 43 |
| 3.1 | THE PROJECT FILES IN PROBLEMS VIEW, SHOWING A LAYOUT FILE WITH A PROBLEM*.* | 44 |
| 3.2 | THE ANDROID STUDIO MAIN WINDOW | 45 |
| 3.3 | AN INLINE VARIABLE VALUE | 51 |
| 3.4 | THE RESULTS OF A LINT INSPECTION IN ANDROID STUDIO | 53 |
| 3.5 | ANDROID STUDIO | 55 |

**ABSTRACT**

The project is aimed to assist farmers with agriculture while conserving valuable resources. An automated rover powered with an intelligent algorithm is designed. Receiving values from multiple sensors placed in the field, the rover will automatically reach the specific area in the field that needs attention. This rover can be used for all agricultural activities like irrigation, fertilization etc., while conserving the resources. A mobile app is provided for the user to monitor the parameters of their field from a remote place. For example, the rover checks for macronutrients in the soil and fertilizes the specific area in the field that requires the nutrients automatically.

**CHAPTER 1: INTRODUCTION**

**1.1 INDUSTRIAL AUTOMATION:**

Instrumentation engineering is the engineering specialization focused on the principle and operation of measuring instruments that are used in design and configuration of automated systems in areas such as electrical and pneumatic domains, and the control of quantities being measured. They typically work for industries with [automated](https://en.wikipedia.org/wiki/Automated) processes, such as [chemical](https://en.wikipedia.org/wiki/Chemical_plant) or [manufacturing](https://en.wikipedia.org/wiki/Manufacturing) plants, with the goal of improving system [productivity](https://en.wikipedia.org/wiki/Productivity), reliability, safety, optimization and stability. To control the parameters in a process or in a particular system, devices such as microprocessors, microcontrollers or PLCs are used, but their ultimate aim is to control the parameters of a system.

Instrumentation engineering is loosely defined because the required tasks are very domain dependent. An expert in the biomedical instrumentation of laboratory rats has very different concerns than the expert in rocket instrumentation. Common concerns of both are the selection of appropriate sensors based on size, weight, cost, reliability, accuracy, longevity, environmental robustness and frequency response. Some sensors are literally fired in artillery shells. Others sense thermonuclear explosions until destroyed. Invariably sensor data must be recorded, transmitted or displayed. Recording rates and capacities vary enormously. Transmission can be trivial or can be clandestine, encrypted and low-power in the presence of jamming. Displays can be trivially simple or can require consultation with [human factors](https://en.wikipedia.org/wiki/Human_factors) experts. Control system design varies from trivial to a separate specialty.

Instrumentation engineers are responsible for integrating the sensors with the recorders, transmitters, displays or control systems, and producing the [Piping and instrumentation diagram](https://en.wikipedia.org/wiki/Piping_and_instrumentation_diagram) for the process. They may design or specify installation, wiring and signal conditioning. They may be responsible for calibration, testing and maintenance of the system.

In a research environment it is common for subject matter experts to have substantial instrumentation system expertise. An astronomer knows the structure of the universe and a great deal about telescopes – optics, pointing and cameras (or other sensing elements). That often includes the hard-won knowledge of the operational procedures that provide the best results. For example, an astronomer is often knowledgeable of techniques to minimize temperature gradients that cause air turbulence within the telescope.

Instrumentation technologists, technicians and mechanics specialize in troubleshooting, repairing and maintaining instruments and instrumentation systems.



*1.0 INDUSTRIAL AUTOMATION*

**1.2 IOT:**

Internet of Things is the concept of connecting any device (so long as it has an on/off switch) to the Internet and to other connected devices. The IoT is a giant network of connected things and people – all of which collect and share data about the way they are used and about the environment around them. The goal of the Internet of Things is to enable things to be connected anytime, anyplace, with anyone ideally using any path /network and any service. Devices and objects with built in sensors are connected to an Internet of Things platform, which integrates data from the different devices and applies analytics to share the most valuable information with applications built to address specific needs.

These powerful IoT platforms can pinpoint exactly what information is useful and what can safely be ignored. This information can be used to detect patterns, make recommendations, and detect possible problems before they occur.



*1.1 IOT*

**1.3 ROVER ROBOTICS:**

A rover is a planetary surface exploration device designed to move across the solid surface on a planet or other planetary celestial bodies. But here we use rover for the agriculture benefits. Rover uses a simple UART protocol which makes them compatible with almost every processor and micro controller. By using this source packages, we offer fast response times to issues and pull requests.



*1.2 ROVER ROBOTICS*

**1.4 CONVENTIONAL AGRICULTURE METHODS:**

The prevailing agricultural system, variously called “conventional farming,” “modern agriculture,” or “industrial farming,” .In agriculture, there are certain parameters to be considered such as the type of crop, properties of soil, climate etc. Depending upon these parameters, farmers decide which crop is to be cultivated at what time of the year and place. Moreover, to yield a high-quality product, suitable soil, climate and season are not sufficient. It requires a set of procedures which needed to be followed. The measures which are followed to raise crops are called agricultural practices.

* Before raising a crop, the soil in which it is to be grown is prepared by ploughing, leveling , and manuring. Ploughing is the process of loosening and digging of soil using a plough. This helps in proper aeration of the soil. After ploughing, the soil is distributed evenly and levelled in the process called levelling. The soil is then manured.
* Selection of seeds of good quality crop strains is the primary stage of sowing. After the preparation of soil, these seeds are dispersed in the field and this is called sowing. Sowing can be done manually, by hand or by using seed drilling machines. Some crops like paddy are first grown into seedlings in a small area and then transplanted to the main field.
* Crops need nutrients to grow and produce yield. Thus, the supply of nutrients at regular intervals is necessary. Manuring is the step where nutritional supplements are provided and these supplements may be natural (manure) or chemical compounds (fertilizers).
* Irrigation is the supply of water. Sources of water can be wells, ponds, lakes, canals, dams etc. Over irrigation may lead to waterlogging and damage the crop. This frequency and interval between successive irrigation need to be controlled.
* Weeds are unwanted plants which grow among crops. They are removed by using weedicides, by manually pulling them with hands and some are removed during soil preparation.
* Once the crop is matured, it is cut and gathered, this process is called harvesting. Followed by harvesting, grains are separated from the chaff either by threshing, or manually in small scale (winnowing).
* Grains yielded are stored in granaries or bins at godowns for later use or marketing.



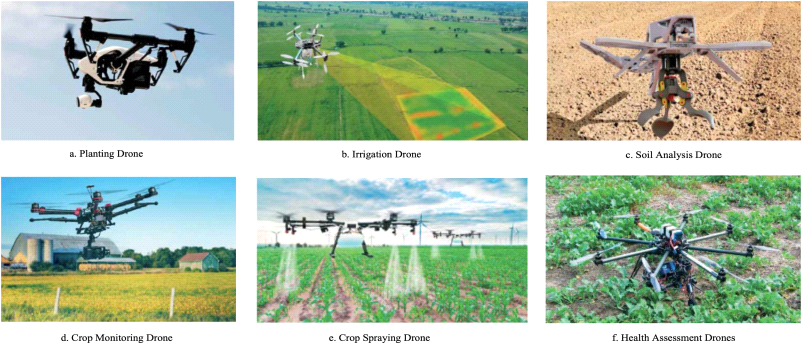
*1.3 CONVETIONAL AGRICULTURE*

**1.5 AUTOMATION IN AGRICULTURE:**

Farm automation (or smart farming) is a variety of tech innovations in traditional farming to optimize the food production process and improve quality. As of now, advanced farming technology can be an essential part of the farmer’s daily work.Now agricultural robots (or “agrobots”) cope with a wide range of tasks: harvesting, watering, seeding, etc. Let’s take a closer look at their work.

Robot-assisted irrigation systems roughly contain two big parts: a Subsurface Drip Irrigation (SDI) system plus special sensors. SDI is well-known in the agriculture industry – it provides an accurate means of controlling the amount of water that is used, and at what time the plants receive it. Even though these systems are obviously more advanced than manual plant-by-plant watering, they are still not perfect as they require some human assistance.

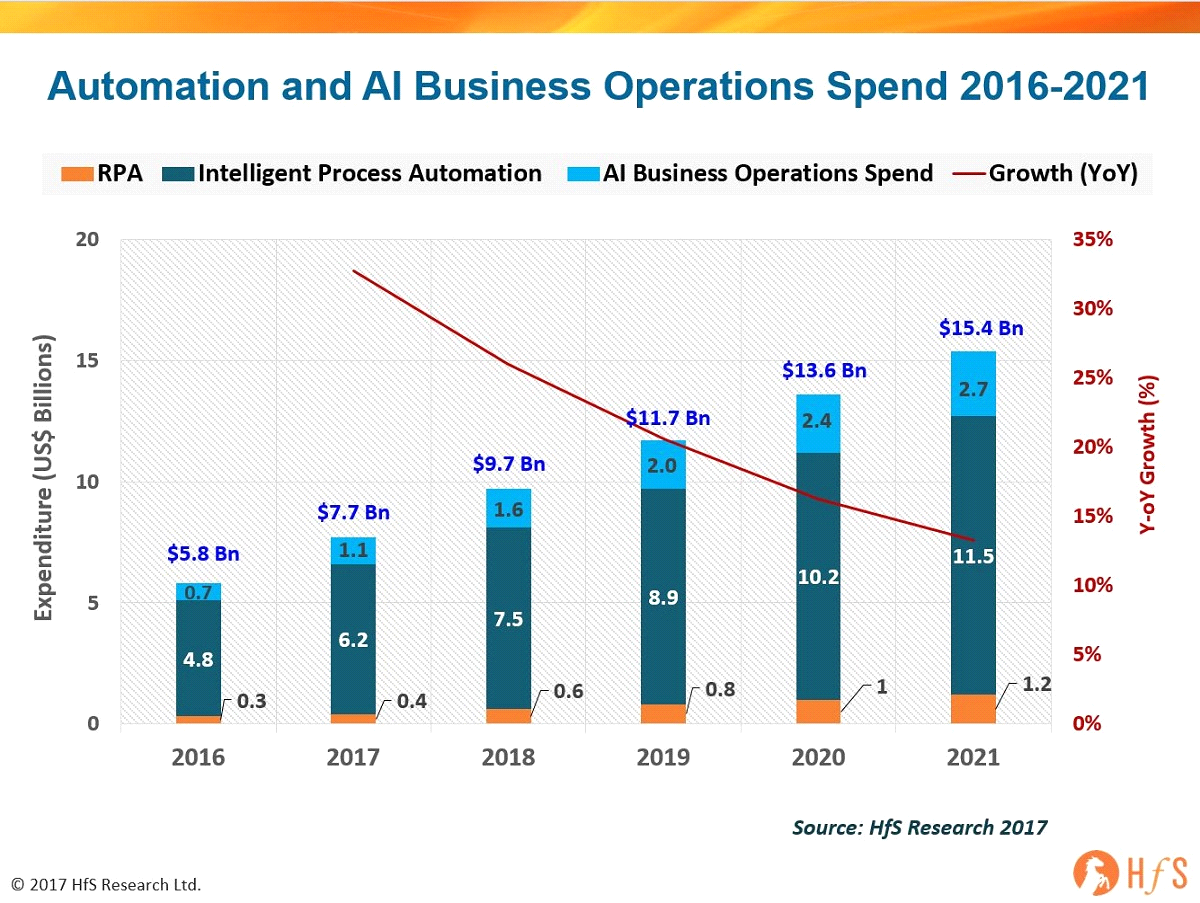
This automation receives automated alerts and remotely investigate and triage an issue, reducing the need to send resources into the field. It also monitors all live and historical media and telemetry date collected from the fleet and reacts quickly with efficient tools to debug and root cause issues remotely.



*1.4 AUTOMATION IN AGRICULTURE*

**CHAPTER 2 : INTERPRETATION SURVEY AND ANALYSIS**

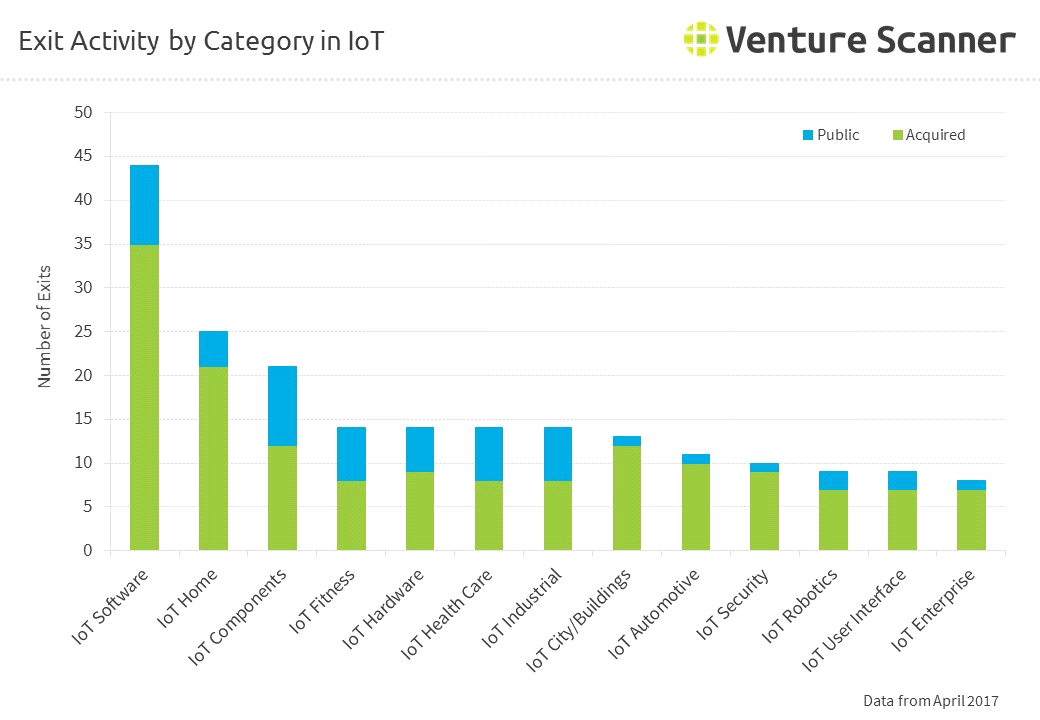
**2.1 INDUSTRIAL AUTOMATION:**



*1.5INDUSTRIAL AUTOMATION GRAPH*

From the above graph it is observed that Automation and AI business market has been growing steadily. The market has been hiked from 5.8 Bn to 15.4 Bn since 2016, showing great potential for developers and engineers who plan to explore the field

**2.2 IOT:**

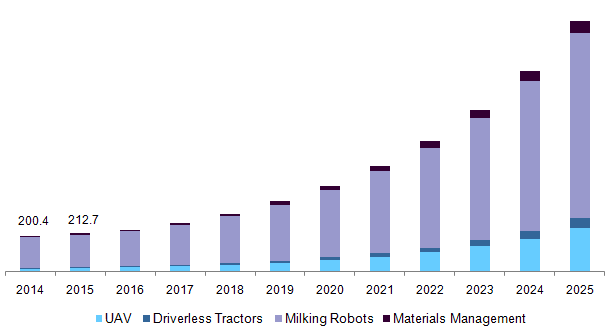


*1.6 IOT GRAPH*

The IoT domain has been experiencing an exponential growth since the boom of the internet. It showed promising results in many industries for remote operations and monitoring while assisting industrial automation. The graph shows the broad market for IoT in various domains.

**2.3 ROVER ROBOTICS:**

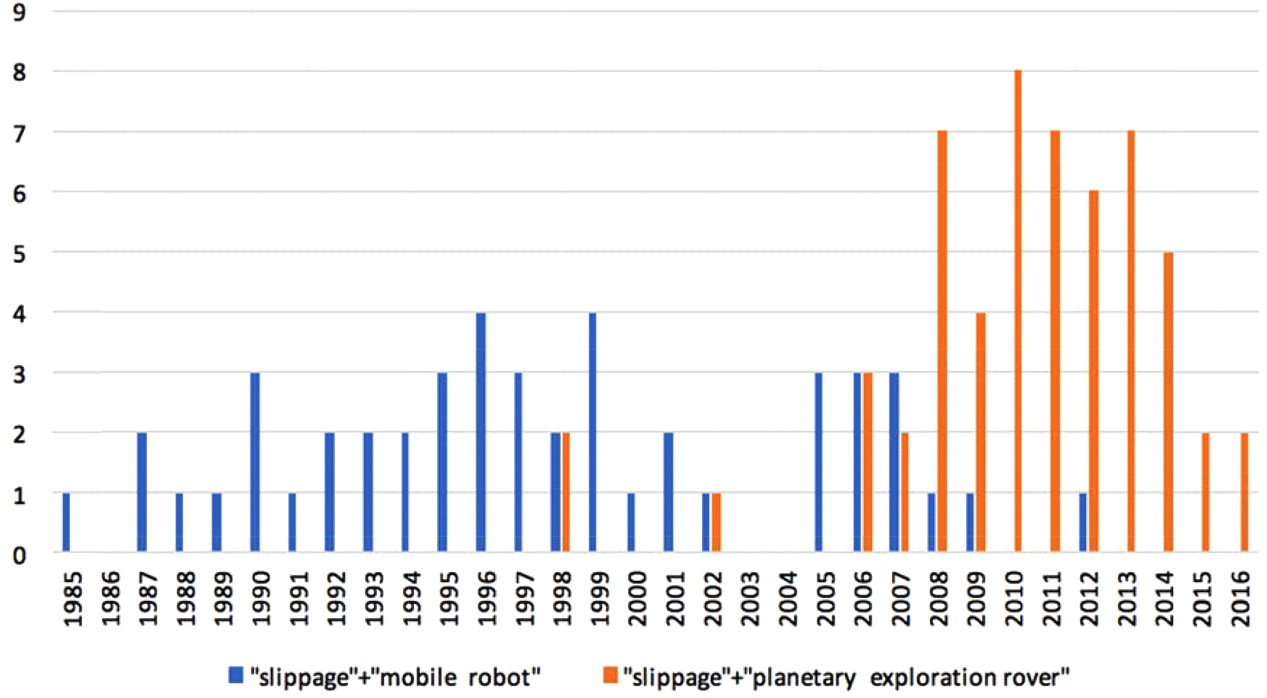
Rover robotics has been implemented in various fields from Space missions to rescue missions. Rovers have been useful in observing and analyzing environments that are hazardous and those that cannot be reached under normal circumstances. Rovers can also be used for domestic purposes such as in an industry or in a mine field. From the below graph we can observe the promising growth of the Rover robotics field in the coming years.



*1.7ROVER ROBOTICS GRAPH*

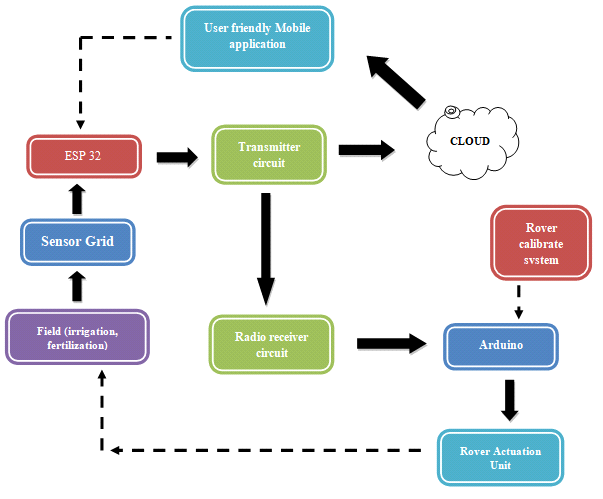
**2.4 AUTOMATION IN AGRICULTURE:**

With the increasing difficulties in agricultural practices many have switched to modern or automation in agriculture. Automation proving to be profitable and conserving valuable resources than in conventional agriculture it shows great potential for future implementations and investments. By analyzing the below graph we can arrive at a point that automation has been adapted by many farmers during the past 6 – 7 years and will keep increasing in the coming future.



*1.8 AUTOMATION IN AGRICULTURE GRAPH*

**CHAPTER 3: BLOCK DIAGRAM**



*1.9 BLOCK DIAGRAM*

**3.1 DESCRIPTION:**

* The sensor grid is formed using moisture sensors and it transmits the value to the ESP32.
* The ESP32 then transmits the values to the cloud and the radio transmitter simultaneously.
* The radio trans-receiver in the ESP32 transmits the value to the radio trans-receiver in the Arduino.
* The Arduino controls the rover actuation unit based on the values received which then controls the parameters of the field.
* The rover calibration unit helps to calibrate the position of the rover in the field.
* The values sent to the cloud is then displayed in the user-friendly mobile application for user’s reference.
* The user can also control the rover using the mobile application.

**3.2RESULT AND DISCUSSION:**

**A plant in a pot

Description automatically generated with medium confidenceGraphical user interface, application

Description automatically generated**

**A plant in a cup

Description automatically generated with low confidenceGraphical user interface, application

Description automatically generated**

**A picture containing wall, indoor, flower, plant

Description automatically generatedGraphical user interface, application

Description automatically generated**

**Graphical user interface, application

Description automatically generated**

**CHAPTER 4: LITERATURE SURVEY**

* 1. **EXISTING SYSTEMS:**

1. **TITLE:** AUTONOMOUS NAVIGATION WITH ROS FOR A MOBILE ROBOT IN AGRICULTURAL FIELDS

**AUTHOR:** Mark A. Post, Alessandro Bianco and Xiu T. Yan

**YEAR:** 2017

**ABSTRACT:** Autonomous monitoring of agricultural farms and fields has recently become feasible due to continuing advances in robotics technology, but many notable challenges remain. In this paper, we describe the state of ongoing work to create a fully autonomous ground rover platform for monitoring and intervention tasks on modern farms that is built using inexpensive and off the shelf hardware and Robot Operating System (ROS) software so as to be affordable to farmers. The hardware and software architectures used in this rover are described along with challenges and solutions in odometry and localization, object recognition and mapping, and path planning algorithms under the constraints of the current hardware. Results obtained from laboratory and field testing show both the key challenges to be overcome, and the current successes in applying a low-cost rover platform to the task of autonomously navigating the outdoor farming environment.

# TITLE: AERIAL FARM ROBOT SYSTEM FOR CROP DUSTING, PLANTING, FERTILIZING AND OTHER FIELD JOBS

**AUTHOR:** Harm BUREMA and Anatoly FILIN

**YEAR**: 2014

**ABSTRACT:** Modern farming is currently being done by powerful ground equipment or aircraft that weigh several tons and treat uniformly tens of hectares per hour. Automated farming can use small, agile, lightweight, energy-efficient automated robotic equipment that flies to do the same job, even able to farm on a plant-by-plant basis, allowing for new ways of farming. Automated farming uses unmanned aerial vehicles (UAVs) that are equipped with detachable implements and reservoirs and that we call "aerial farm robots." Automated farming uses high-precision GPS and other precision positioning and vision technology to autonomously and precisely perform crop dusting, planting, fertilizing and other field related farming or husbandry tasks. The subsystems for the control, refill, recharge and communication subsystems of the aerial farm robots are part of the overall automated farming system, and can autonomously handle most of the husbandry tasks on a farm.

1. **TITLE:** FLEXIBLE AGRICULTURAL AUTOMATION

**AUTHOR:** Brian P. Hanley

**YEAR:** 2002

**ABSTRACT:** Agricultural operations by applying flexible manufacturing software, robotics and sensing techniques to agriculture. In manufacturing operations utilizing flexible machining and flexible assembly robots, work pieces flow through a fixed set of workstations on an assembly line. At different stations are located machine vision systems, laser based raster devices, radar, touch, photocell, and other methods of sensing; flexible robot armatures and the like are used to operate on them. This flexible agricultural automation turns that concept inside out, moving software programmable workstations through farm fields on mobile robots that can sense their environment and respond to it flexibly. The agricultural automation will make it possible for large scale farming to take up labor intensive farming practices which are currently only practical for small scale farming, improving land utilization efficiency, while lowering manpower costs dramatically.

1. **TITLE:** AGRICULTURAL ROBOT SYSTEM AND METHOD

**AUTHOR:** Harvey Koselka and Bret Wallach

**YEAR:** 2006

**ABSTRACT:** An agricultural robot system and method of harvesting, pruning, culling, weeding, measuring and managing of agricultural crops. Uses autonomous and semi-autonomous robot(s) comprising machine-vision using cameras that identify and locate the fruit on each tree, points on a vine to prune, etc., or may be utilized in measuring agricultural parameters or aid in managing agricultural resources. The cameras may be coupled with an arm or other implement to allow views from inside the plant when performing the desired agricultural function. A robot moves through a field first to “map” the plant locations, number and size of fruit and approximate positions of fruit or map the cordons and canes of grape vines. Once the map is complete, a robot or server can create an action plan that a robot may implement. An action plan may comprise operations and data specifying the agricultural function to perform.

# TITLE: SMART IRRIGATION WITH FIELD PROTECTION AND CROP HEALTH MONITORING SYSTEM USING AUTONOMOUS ROVER

**AUTHOR:** S. Gobhinath, M. Devi Darshini, K. Durga, R. Hari Priyanga

**YEAR:** 2019

**ABSTRACT:**Automation has maneuvered industrial advancements and has become provident in most of the domain. But the contribution of automation to agriculture is in a lower degree. So, by incorporating automation in the field of agriculture, productivity can be increased to manifolds. This paper brings out the ways to automate agriculture which enhances irrigation, protection of farmlands and health management. Irrigation of crops is made economical by considering the moisture level in soil, temperature and humidity. Health supervision is done by an autonomous agricultural rover which moves around the field, collecting data through a camera fixed on it. The images are processed using algorithms in MATLAB to identify the disease affecting or nutrition deficit in the field. The shortcoming is indicated to the farmer. In case of fire accidents, it is detected using Ultraviolet Flame sensor and the fire is put off. Also, the farm is protected from animal intrusion using PIR sensor and buzzer. These techniques are integrated to improve the standards of agricultural farming.

# TITLE:SIMPLIFIED MODELLING AND BACKSTEPPING CONTROL OF THE LONG ARM AGRICULTURAL ROVER

**AUTHOR:** NapasoolWongvanich, SungwanBoksuwan&AbdulhafizChesof.

**YEAR:** 2020

**ABSTRACT:** This paper presents the development of the simplified modelling and control of a long arm system for an agricultural rover, which also extends the modelling methodology from the previous work. The methodology initially assumes a flexible model and, through the use of the integral-based parameter identification method, the identified parameters are then correlated to an energy function to allow a construction of the friction induced nonlinear vibration model. To also capture the effect of the time delay, a delay model was also considered in the form of a second order delay differential equation. Both families of models were applied to identify and characterise a specialised long arm system. The nonlinear model was found to give significant improvement over the standard linear model in data fitting, which was further enhanced by the addition of the time delay consideration. A backstepping controller was also designed for both model families. Results show that the delay model expends less control efforts than the lesser non-delay model.

## ADVANTAGES OVER EXISTING SYSTEMS:

* Existing systems are implemented only for specific processes the proposed project can perform multiple agricultural processes.
* Existing irrigation systems use a centralized algorithm which in case of this project, it monitors specific areas in the field individually which reduces the water and other resources usage to a significant amount.
* The proposed system is mobile which also helps in keeping out birds and other pests which is not possible in the prevailing stationery systems.

**CHAPTER 5: AUTOMATED ROVER**

**5.1 PROPOSED SYSTEM:**

* A grid is drawn over the field dividing it into different partitions. In each partition a REES52 soil moisture sensor is placed which is connected to an ESP32 placed at a stationery point in a field.
* The readings from the sensors are collected in the ESP32 and based on the algorithm developed the ESP32 transmits various commands to the Arduino placed in the rover through the nRF24l01 + radio transmitter present which can transmit data to a range of 60 meters.
* A sound source of certain frequency is placed at one of the ends of the field to calibrate the position of the rover. When a sensor at a particular position in the grid detects the low level of moisture, it is transmitted to the rover reaches the point and irrigates water till required moisture level is reached.
* A mobile is developed to enable the user to manually monitor the parameters of their field and control the rover remotely using Internet Of Things when connected to the internet. Using the application the user can instruct the rover to initiate fertilization.
* The rover will inspect the areas for nutrients like Nitrogen, Potassium and Phosphorous using an optical transducer TCS230 equipped in the rover and it will fertilize the field.

**5.2 REQUIREMENTS:**

**HARDWARE REQUIREMENTS:**

* Arduino Uno
* ESP32
* L293D motor shield for Arduino
* REES52 Soil Moisture Sensor
* NRF24l01 + Radio trans receiver module
* Buzzer and Mic module
* TCS230 colour sensor
* 9V DC motor
* 12V Lead acid battery
* Water pump 60psi output
* Water Sprinkler Head

**SOFTWARE REQUIREMENTS:**

* Arduino IDE
* Android Studio

**CHAPTER 6 : HARDWARE DESCRIPTION**

## 6.1 ARDUINO UNO:

Arduino Uno is a microcontroller board based on the ATmega328P . It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz ceramic resonator , 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.

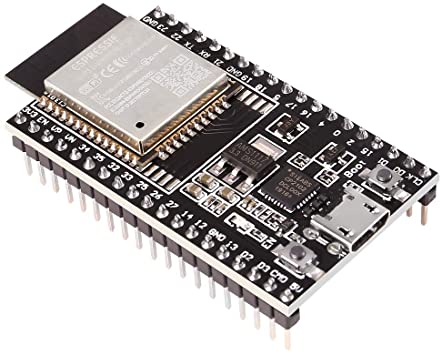


*2.0 ARDUINO UNO*

## 6.2 ESP32:

ESP32 is capable of functioning reliably in industrial environments, with an operating temperature ranging from –40°C to +125°C. Powered by advanced calibration circuitries, ESP32 can dynamically remove external circuit imperfections and adapt to changes in external conditions. ESP32 is highly-integrated with in-built antenna switches, RF balun, power amplifier, low-noise receive amplifier, filters, and power management modules. ESP32 adds priceless functionality and versatility to your applications with minimal Printed Circuit Board (PCB) requirements. ESP32 can perform as a complete standalone system or as a slave device to a host MCU, reducing communication stack overhead on the main application processor. ESP32 can interface with other systems to provide Wi-Fi and Bluetooth functionality through its SPI / SDIO or I2C / UART interface.

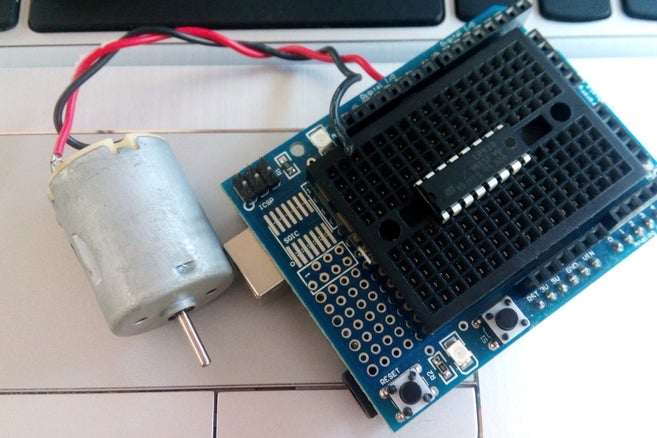
.



*2.1 ESP32*

## 6.3 L293D MOTOR SHIELD FOR ARDUINO:

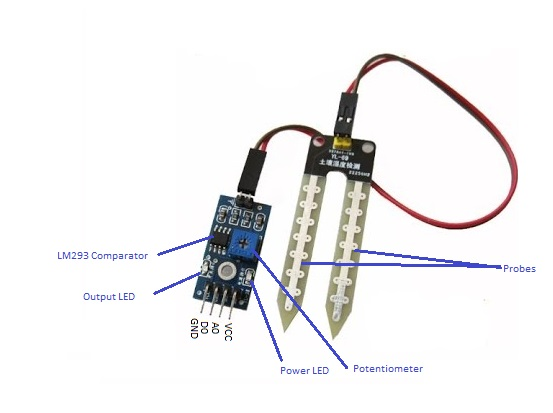
The L293D is a 16-pin Motor Driver IC which can control a set of two DC motors simultaneously in any direction. The L293D is designed to provide bidirectional drive currents of up to 600 mA (per channel) at voltages from 4.5 V to 36 V (at pin 8). You can use it to control small dc motors - toy motors.



*2.2 L293D MOTOR DRIVE*

**6.4 REES52 SOIL MOISTURE SENSOR:**

It easy to use digital soil moisture sensor. Just insert the sensor in the soil and it can measure moisture or water level content in it. It gives a digital output of 5V when the moisture level is high and 0V when the moisture level is low in the soil. The sensor includes a potentiometer to set the desired moisture threshold. When the sensor measures more moisture than the set threshold, the digital output goes high and an LED indicates the output. When the moisture in the soil is less than the set threshold, the output remains low. The digital output can be connected to a microcontroller to sense the moisture level. The sensor also outputs an analog output which can be connected to the ADC of a microcontroller to get the exact moisture level in the soil. This sensor is great for making water gardening projects, water sensing, etc.

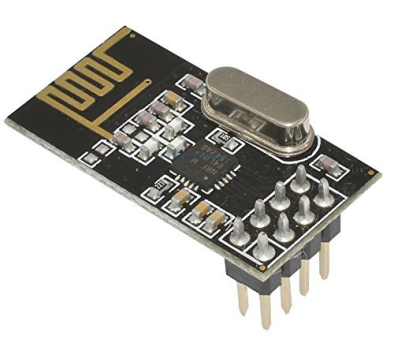


*2.3 SOIL MOISTURE SENSOR*

## 6.5 NRF24l01 + RADIO TRANSRECEIVER MODULE:

Having two or more Arduino boards be able to communicate with each other wirelessly over a distance opens lots of possibilities like remotely monitoring sensor data, controlling robots, home automation and the list goes on. And when it comes down to having inexpensive yet reliable 2-way RF solutions, no one does a better job than nRF24L01+ transceiver module from [Nordic Semiconductor](http://www.nordicsemi.com/). The nRF24L01+ transceiver module is designed to operate in 2.4 GHz worldwide ISM frequency band and uses [GFSK modulation](https://en.wikipedia.org/wiki/Frequency-shift_keying#Gaussian_frequency-shift_keying) for data transmission. The data transfer rate can be one of 250kbps, 1Mbps and 2Mbps.

The operating voltage of the module is from 1.9 to 3.6V, but the good news is that the logic pins are 5-volt tolerant, so we can easily connect it to an Arduino or any 5V logic microcontroller without using any logic level converter. The nRF24L01+ transceiver module communicates over a 4-pin Serial Peripheral Interface (SPI) with a maximum data rate of 10Mbps. All the parameters such as frequency channel (125 selectable channels), output power (0 dBm, -6 dBm, -12 dBm or -18 dBm), and data rate (250kbps, 1Mbps, or 2Mbps) can be configured through SPI interface.

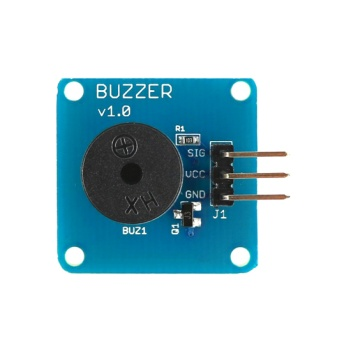


*2.4 NRF24L01 + RADIO TRANS RECEIVER MODULE*

## 6.6 BUZZER AND MIC MODULE:

An Active Buzzer Alarm Module for Arduino is an audio signaling device, which may be mechanical, electromechanical, or piezoelectric. The buzzer consists of an outside case with two pins to attach it to power and ground.Inside is a piezo element, which consists of a central ceramic disc surrounded by a metal (often bronze)vibration disc.

When current is applied to the buzzer it causes the ceramic disk to contract or expand and by Changing this then causes the surrounding disc to vibrate.  That’s the sound that you hear.  By changing the frequency of the buzzer, the speed of the vibrations changes, which changes the pitch of the resulting sound.



*2.5 BUZZER AND MIC MODULE*

## 6.7 TCS230 COLOUR SENSOR:

The TCS3200 color sensor can detect a wide variety of colors based on their wavelength. This sensor is speciallyuseful for color recognition projects such as color matching, color sorting, test strip reading. The TCS3200 has an array of photodiodes with 4 different filters. A photodiode is simply a semiconductor device that converts light into current. The sensor has a current-to-frequency converter that converts the photodiodes’ readings into a square wave with a frequency that is proportional to the light intensity of the chosen color. This frequency is then, read by the Arduino.



*2.6 TCS230 COLOUR SENSOR*

## 6.8 9v DC MOTOR:

9V DC Motor is a great choice for projects that require high speed and torque. It can be used in robotics projects as well as industrial applications. Nearly all types of DC motors have some internal mechanism, either electromechanical or electronic, to periodically change the direction of current in part of the motor. DC motor's speed can be controlled over a wide range, using either a variable supply voltage or by changing the strength of current in its field windings.



*2.7 9v DC MOTOR*

## 6.9 12v LEAD ACID BATTERY:

* Lead-acid is the oldest rechargeable battery in existence. Lead-acid does not lend itself to fast charging. Typical charge time is 8 to 16 hours. A periodic fully saturated charge is essential to prevent sulfation and the battery must always be stored in a charged state. Leaving the battery in a discharged condition causes sulfation and a recharge may not be possible.  A high voltage (above 2.40V/cell) produces good battery performance but shortens the service life due to grid corrosion on the positive plate. A low voltage limit is subject to sulfation on the negative plate. Leaving the battery on float charge for a prolonged time does not cause damage.
* To prevent the battery from being stressed through repetitive deep discharge, a larger battery is recommended. Lead-acid is inexpensive but the operational costs can be higher than a nickel-based system if repetitive full cycles are required.
* Depending on the depth of discharge and operating temperature, the sealed lead-acid provides 200 to 300 discharge/charge cycles. The primary reason for its relatively short cycle life is grid corrosion of the positive electrode, depletion of the active material and expansion of the positive plates. These changes are most prevalent at higher operating temperatures.
* Sealed lead acid batteries can have a design life of anywhere from 3 – 5 years all the way up to 12+ years depending on the manufacturing process of the battery.



*2.8 12v LEAD ACID BATTERY*

# CHAPTER 7 : SOFTWARE DESCRIPTION

## 7.1 ARDUINO IDE:

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

**7.1.1 WRITING SKETCHES**

Programs written using Arduino Software (IDE) are called sketches. These sketches are written in the text editor and are saved with the file extension .ino. The editor has features for cutting/pasting and for searching/replacing text. The message area gives feedback while saving and exporting and also displays errors. The console displays text output by the Arduino Software (IDE), including complete error messages and other information. The bottom right hand corner of the window displays the configured board and serial port. The toolbar buttons allow you to verify and upload programs, create, open, and save sketches, and open the serial monitor. NB: Versions of the Arduino Software (IDE) prior to 1.0 saved sketches with the extension .pde. It is possible to open these files with version 1.0, you will be prompted to save the sketch with the .ino extension on save.

**Verify**

Checks your code for errors compiling it.

**Upload**

Compiles your code and uploads it to the configured board. See uploading below for details. Note: If you are using an external programmer with your board, you can hold down theshiftkey on your computer when using this icon. The text will change to &quot;upload using Programmer&quot;

**New**

Creates a new sketch.

**Open**

Presents a menu of all the sketches in your sketchbook. Clicking one will open it Within the current window overwriting its content. Note: due to a bug in Java, this menu scroll if you need to open a sketch late in the list, use the File | Sketchbookmenu instead.

**Save**

Saves your sketch.

**Serial Monitor**

Opens the serial monitor. Additional commands are found within the five menus: File, Edit, Sketch, Tools, and Help. The menus are context sensitive, which means only those items relevant to the work currently being carried out are available.

**7.1.2 FILE**

**New**

Creates a new instance of the editor, with the bare minimum structure of a sketch already in place.

**Open**

Allows loading a sketch file browsing through the computer drives and folders.

**Open Recent**

Provides a short list of the most recent sketches, ready to be opened.

**Sketchbook**

Shows the current sketches within the sketchbook folder structure; clicking on any name opens the corresponding sketch in a new editor instance.

**Examples**

Any example provided by the Arduino Software (IDE) or library shows up in this menu item. All the examples are structured in a tree that allows easy access by topic or library.

**Close**

Closes the instance of the Arduino Software from which it is clicked.

**Save**

Saves the sketch with the current name. If the file hasn’t been named before, a name will be provided in “Save as” window.

**Save as...**

Allows saving the current sketch with a different name.

**Page Setup**

It shows the Page Setup window for printing.

**Print**

Sends the current sketch to the printer according to the settings defined in Page Setup.

**Preferences**

Opens the Preferences window where some settings of the IDE may be customized, as the language of the IDE interface.

**Quit**

Closes all IDE windows. The same sketches open when Quit was chosen will be

Automatically reopened the next time you start the IDE.

**7.1.3 EDIT**

**Undo/Redo**

Goes back of one or more steps you did while editing; when you go back, you may go forward with Redo.

**Cut**

Removes the selected text from the editor and places it into the clipboard.

**Copy**

Duplicates the selected text in the editor and places it into the clipboard.

**Copy for Forum**

Copies the code of your sketch to the clipboard in a form suitable for posting to the forum, complete with syntax colouring.

**Copy as HTML**

Copies the code of your sketch to the clipboard as HTML, suitable for embedding in web pages.

**Paste**

Puts the contents of the clipboard at the cursor position, in the editor.

**Select All**

Selects and highlights the whole content of the editor.

**Comment/Uncomment**

Puts or removes the // comment marker at the beginning of each selected line.

**Increase/Decrease Indent**

Adds or subtracts a space at the beginning of each selected line, moving the text one space on the right or eliminating a space at the beginning.

**Find**

Opens the Find and Replace window where you can specify text to search inside the current sketch according to several options.

**Find Next**

Highlights the next occurrence - if any - of the string specified as the search item in the

Find window, relative to the cursor position.

**Find Previous**

Highlights the previous occurrence - if any - of the string specified as the search item in the Find window relative to the cursor position.

**7.1.4 SKETCH**

**Verify/Compile**

Checks your sketch for errors compiling it; it will report memory usage for code and variables in the console area.

**Upload**

Compiles and loads the binary file onto the configured board through the configured Port.

**Upload Using Programmer**

This will overwrite the boot loader on the board; you will need to use Tools &gt; Burn Boot loader to restore it and be able to Upload to USB serial port again. However, it allows you to use the full capacity of the Flash memory for your sketch. Please note that this command will NOT burn the fuses. To do so Tools -> Burn Boot loader command must be executed.

**Export Compiled Binary**

Saves a .hex file that may be kept as archive or sent to the board using other tools.

**Show Sketch Folder**

Opens the current sketch folder.

**Include Library**

Adds a library to your sketch by inserting #include statements at the start of your code. For more details, seelibraries below. Additionally, from this menu item you can access the Library Manager and import new libraries from .zip files.

**Add File...**

Adds a source file to the sketch (it will be copied from its current location). The new file appears in a new tab in the sketch window. Files can be removed from the sketch using the tab menu accessible clicking on the small triangle icon below the serial monitor one on the right side o the toolbar.

**7.1.5 TOOLS**

**Auto Format**

This formats your code nicely: i.e. indents it so that opening and closing curly braces line up, and that the statements inside curly braces are indented more.

**Archive Sketch**

Archives a copy of the current sketch in .zip format. The archive is placed in the same directory as the sketch.

**Fix Encoding &amp; Reload**

Fixes possible discrepancies between the editor char map encoding and other operating systems char maps.

**Serial Monitor**

Opens the serial monitor window and initiates the exchange of data with any connected board on the currently selected Port. This usually resets the board, if the board supports Reset over serial port opening.

**Board**

Select the board that your using. See below for descriptions of the various boards.

**Port**

This menu contains all the serial devices (real or virtual) on your machine. It should automatically refresh every time you open the top-level tools menu.

**Programmer**

For selecting a hardware programmer when programming a board or chip and not using the on-board USB-serial connection. Normally you won’t need this, but if you’re burning a bootloader to a new microcontroller, you will use this.

**Burn Boot loader**

The items in this menu allow you to burn a boot loader onto the microcontroller on an Arduino board. This is not required for normal use of an Arduino or Genuino board but is useful if you purchase new AT mega microcontrollers (which normally come without a boot loader). Ensure that you have selected the correct board from the Boards menu before burning the bootloader on the target board. This command also set the right fuses.

**7.1.6 HELP**

Here you find easy access to a number of documents that come with the Arduino Software(IDE). You have access to Getting Started, Reference, this guide to the IDE and other documents locally, without an internet connection. The documents are a local copy of the online ones and may link back to our online website.

**Find in Reference**

This is the only interactive function of the Help menu: it directly selects the relevant page in the local copy of the Reference for the function or command under the cursor.

**7.1.7 SKETCHBOOK**

The Arduino Software (IDE) uses the concept of a sketchbook: a standard place to store your programs (or sketches). The sketches in your sketchbook can be opened from the File >Sketchbook menu or from the Open button on the toolbar. The first time you run the Arduino software, it will automatically create a directory for your sketchbook. You can view or change the location of the sketchbook location from with the Preferences dialog.

Beginning with version 1.0, files are saved with a .ino file extension. Previous versions use the .pde extension. You may still open .pde named files in version 1.0 and later, the software will automatically rename the extension to .ino.

**7.2TABS, MULTIPLE FILES, AND COMPILATION**

Allow you to manage sketches with more than one file (each of which appears in its own tab). These can be normal Arduino code files (no visible extension), C files (.c extension), C++files (.cpp), or header files (.h).

**7.2.1 UPLOADING**

Before uploading your sketch, you need to select the correct items from the Tools > Board and Tools >Port menus. The boards are described below. On the Mac, the serial port is probably something like /dev/tty.usbmodem241 (for a Uno or Mega2560 or Leonardo) or /dev/tty.usbserial-1B1 (for a Duemilanove or earlier USB board), or/dev/tty.USA19QW1b1P1.1 (for a serial board connected with a Key span USB-to-Serial adapter). On Windows, it&#39;s probably COM1 or COM2 (for a serial board) or COM4, COM5, COM7, or higher (for a USB board) - to find out, you look for USB serial device in the port section of the Windows Device Manager. On Linux, it should be /dev/ttyACMx, /dev/ttyUSBx or similar.

Once you’ve selected the correct serial port and board, press the upload button in the toolbar or select the Upload item from the File menu. Current Arduino boards will resetautomatically and begin the upload.

With older boards (pre-Diecimila) that lack auto-reset, you will need to press the resetbutton on the board just before starting the upload. On most boards, you will see the RX andTX LEDs blink as the sketch is uploaded. The Arduino Software (IDE) will display a messagewhen the upload is complete, or show an error. When you upload a sketch, you’re using the Arduino bootloader, a small program that has been loaded on to the microcontroller on your board.

It allows you to upload code without using any additional hardware. The boot loader is active for a few seconds when the board resets; then it starts whichever sketch was most recentlyuploaded to the microcontroller. The bootloader will blink the on-board (pin 13) LED when it starts (i.e. when the board resets).

**7.2.2 LIBRARIES**

Libraries provide extra functionality for use in sketches, e.g. working with hardware ormanipulating data. To use a library in a sketch, select it from the Sketch > Import Library menu. This will insert one or more #include statements at the top of the sketch and compile the library with your sketch. Because libraries are uploaded to the board with your sketch, they increase the amount of space it takes up. If a sketch no longer needs a library, simply delete its #include statements from the top of your code. There is a list of libraries in the reference. Some libraries are included with the

Arduino software. Others can be downloaded from a variety of sources or through the Library Manager. Starting with version 1.0.5 of the IDE, you do can import a library from a zip file and use it in an open sketch. See these instructions for installing a third-party library.To write your own library, see this tutorial.

**7.3 THIRD-PARTY HARDWARE**

Support for third-party hardware can be added to the hardware directory of your sketchbook directory. Platforms installed there may include board definitions (which appear in the board menu), core libraries, boot loaders, and programmer definitions.

To install, create the hardware directory, then unzip the third-party platform into its own sub-directory. (Don’t “arduino” as the sub-directory name or yow will override the built-in Arduino platform.) To uninstall, simply delete its directory. For details on creating packages for third-party hardware, see the Arduino IDE 1.5 3rd party Hardware specification.

**7.3.1 SERIAL MONITOR**

Displays serial data being sent from the Arduino or Genuino board (USB or serial board). To send data to the board, enter text and click on the ”send” button or press enter. Choose thebaud rate from the drop-down that matches the rate passed to Serial. Begin in your sketch. Note that on Windows, Mac or Linux, the Arduino or Genuino board will reset (rerun your sketch execution to the beginning) when you connect with the serial monitor. You can also talk to the board from Processing, Flash, MaxMSP, etc (see the interfacing page for details).

**7.4 REFERENCES**

Some preferences can be set in the preferences dialog (found under the Arduino menu on the Mac, or File on Windows and Linux). The rest can be found in the preferences file, whose location is shown in the preference dialog.Software (IDE) to update it to the new default language.

**7.5 BOARDS**

The board selection has two effects: it sets the parameters (e.g. CPU speed and baud rate)used when compiling and uploading sketches; and sets and the file and fuse settings used by the burn boot loader command. Some of the board definitions differ only in the latter, so even ifYou’ve been uploading successfully with a particular selection you will want to check it before burning the boot loader. You can find a comparison table between the various boards here.

Arduino Software (IDE) includes the built in support for the boards in the following list, allbased on the AVR Core. The Boards included in the standard installation allows to add support for the growing number of new boards based on different cores like Arduino Due, Arduino Zero, Edison, and Galileo and so on.

* **Arduino**

An ATmega32u4 running at 16 MHz with auto-reset, 12 Analog In, 20 Digital I/O and 7PWM.

* **Arduino/Genuino Uno**

An ATmega328 running at 16 MHz with auto-reset, 6 Analog In, 14 Digital I/O and 6PWM.

* **ArduinoDiecimila or Duemilanove w/ ATmega168**

An ATmega168 running at 16 MHz with auto-reset.

* **Arduino Nano w/ ATmega328**

An ATmega328 running at 16 MHz with auto-reset. Has eight analog inputs.

* **Arduino/Genuino Mega 2560**

An ATmega2560 running at 16 MHz with auto-reset, 16 Analog In, 54 Digital I/O and 15PWM.

* **Arduino Mega**

An ATmega1280 running at 16 MHz with auto-reset, 16 Analog In, 54 Digital I/O and 15PWM.

* **Arduino Mega ADK**

An ATmega2560 running at 16 MHz with auto-reset, 16 Analog In, 54 Digital I/O and 15PWM.

* **Arduino Leonardo**

An ATmega32u4 running at 16 MHz with auto-reset, 12 Analog In, 20 Digital I/O and 7PWM.

* **Arduino Micro**

An ATmega32u4 running at 16 MHz with auto-reset, 12 Analog In, 20 Digital I/O and 7PWM.

* **ArduinoEsplora**

An ATmega32u4 running at 16 MHz with auto-reset.

* **Arduino Mini w/ ATmega328**

An ATmega328 running at 16 MHz with auto-reset, 8 Analog In, 14 Digital I/O and 6PWM.

* **Arduino Ethernet**

Equivalent to Arduino UNO with an Ethernet shield: An ATmega328 running at16 MHz with auto-reset, 6 Analog In, 14 Digital I/O and 6 PWM.

* **ArduinoFio**

An ATmega328 running at 8 MHz with auto-reset. Equivalent to Arduino Pro or ProMini (3.3V, 8 MHz) w/ATmega328, 6 Analog In, 14 Digital I/O and 6 PWM.

* **Arduino BT w/ ATmega328**

ATmega328 running at 16 MHz The boot loader burned (4 KB) includes codes toinitialize the on-board Bluetooth module, 6 Analog In, 14 Digital I/O and 6 PWM..

* **Lily Pad Arduino USB**

An ATmega32u4 running at 8 MHz with auto-reset, 4 Analog In, 9 Digital I/O and 4PWM.

* **Lily Pad Arduino**

An ATmega168 or ATmega132 running at 8 MHz with auto-reset, 6 Analog In, 14Digital I/O and 6 PWM.

* **Arduino Pro or Pro Mini (5V, 16 MHz) ATmega328**

An ATmega328 running at 16 MHz with auto-reset. Equivalent to ArduinoDuemilanoveor Nano w/ ATmega328; 6 Analog In, 14 Digital I/O and 6 PWM.

* **Arduino NG or older  ATmega168**

An ATmega168 running at 16 MHz without auto-reset. Compilation and upload isequivalent to ArduinoDiecimila or Duemilanove w/ ATmega168, but the bootloaderburned has a slower timeout (and blinks the pin 13 LED three times on reset); 6 AnalogIn, 14 Digital I/O and 6 PWM.

* **Arduino Robot Control**

An ATmega328 running at 16 MHz with auto-reset.

* **Arduino Robot Motor**

An ATmega328 running at 16 MHz with auto-reset.

* **Arduino Gemma**

An ATtiny85 running at 8 MHz with auto-reset, 1 Analog In, 3 Digital I/O and 2 PWM.

## C:\Users\rajar\Downloads\Arduino-IDE.png

*2.9 ARDUINO IDE*

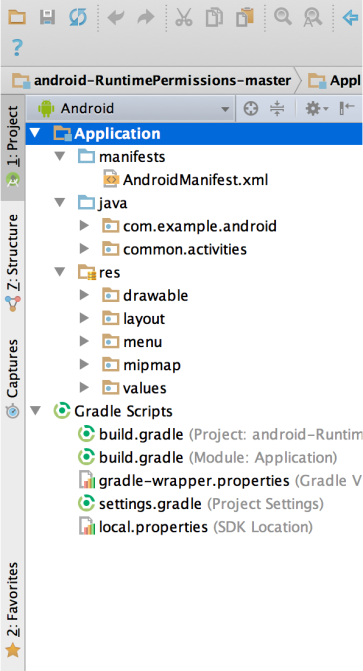
## 7.6 ANDROID STUDIO:

Android Studio is the official Integrated Development Environment (IDE) for Android app development, based on [IntelliJ IDEA](https://www.jetbrains.com/idea/). On top of IntelliJ's powerful code editor and developer tools, Android Studio offers even more features that enhance your productivity when building Android apps, such as:

* A flexible Gradle-based build system
* A fast and feature-rich emulator
* A unified environment where you can develop for all Android devices
* Apply Changes to push code and resource changes to your running app without restarting your app
* Code templates and GitHub integration to help you build common app features and import sample code
* Extensive testing tools and frameworks
* Lint tools to catch performance, usability, version compatibility, and other problems
* C++ and NDK support
* Built-in support for [Google Cloud Platform](https://cloud.google.com/tools/android-studio/docs/), making it easy to integrate Google Cloud Messaging and App Engine

This page provides an introduction to basic Android Studio features. For a summary of the latest changes, see [Android Studio release notes](https://developer.android.com/studio/releases).

**Project structure**



*3.0* *THE PROJECT FILES IN ANDROID VIEW.*

Each project in Android Studio contains one or more modules with source code files and resource files. Types of modules include:

* Android app modules
* Library modules
* Google App Engine modules

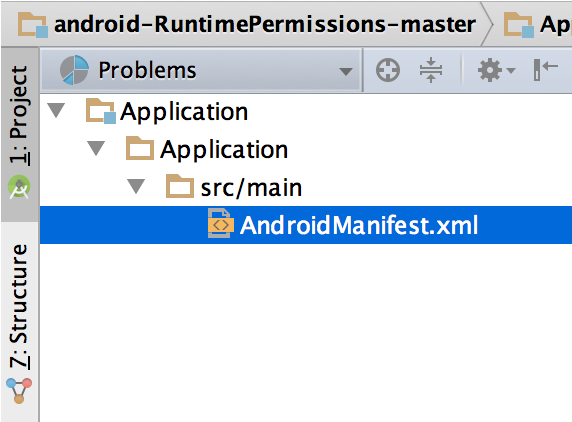
By default, Android Studio displays your project files in the Android project view, as shown in figure 1. This view is organized by modules to provide quick access to your project's key source files.

All the build files are visible at the top level under **Gradle Scripts** and each app module contains the following folders:

* **manifests**: Contains the AndroidManifest.xml file.
* **java**: Contains the Java source code files, including JUnit test code.
* **res**: Contains all non-code resources, such as XML layouts, UI strings, and bitmap images.

The Android project structure on disk differs from this flattened representation. To see the actual file structure of the project, select **Project** from the **Project** dropdown (in figure 3.1, it's showing as **Android**).

You can also customize the view of the project files to focus on specific aspects of your app development. For example, selecting the **Problems** view of your project displays links to the source files containing any recognized coding and syntax errors, such as a missing XML element closing tag in a layout file.

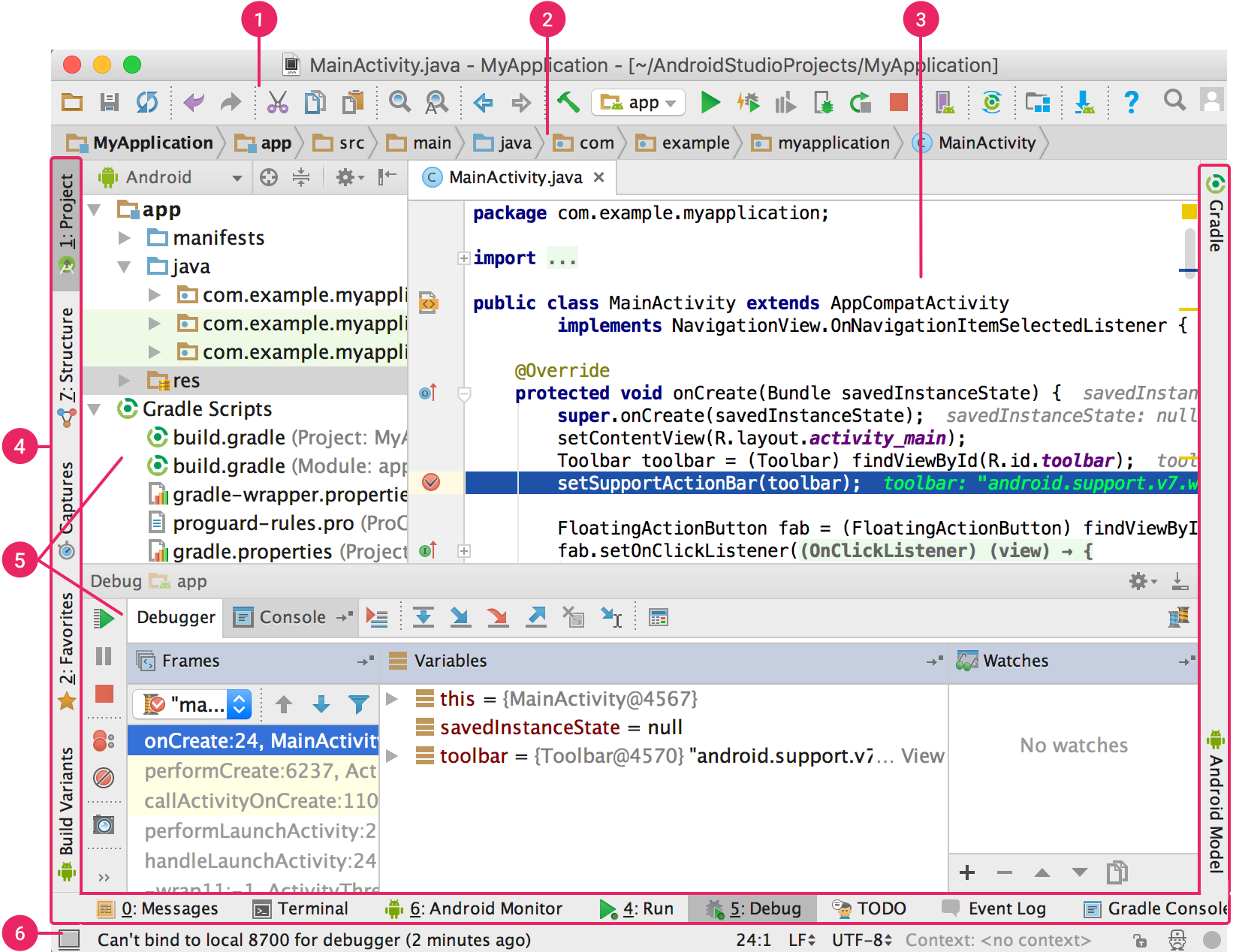


***3.1*** *THE PROJECT FILES IN PROBLEMS VIEW, SHOWING A LAYOUT FILE WITH A PROBLEM.*

For more information, see [Projects overview](https://developer.android.com/studio/projects).

## The user interface

The Android Studio main window is made up of several logical areas identified in figure 3.



*3.2* *THE ANDROID STUDIO MAIN WINDOW.*

1. The **toolbar** lets you carry out a wide range of actions, including running your app and launching Android tools.
2. The **navigation bar** helps you navigate through your project and open files for editing. It provides a more compact view of the structure visible in the **Project** window.
3. The **editor window** is where you create and modify code. Depending on the current file type, the editor can change. For example, when viewing a layout file, the editor displays the Layout Editor.
4. The **tool window bar** runs around the outside of the IDE window and contains the buttons that allow you to expand or collapse individual tool windows.
5. The **tool windows** give you access to specific tasks like project management, search, version control, and more. You can expand them and collapse them.
6. The **status bar** displays the status of your project and the IDE itself, as well as any warnings or messages.

You can organize the main window to give yourself more screen space by hiding or moving toolbars and tool windows. You can also use keyboard shortcuts to access most IDE features.

At any time, you can search across your source code, databases, actions, elements of the user interface, and so on, by double-pressing the Shift key, or clicking the magnifying glass in the upper right-hand corner of the Android Studio window. This can be very useful if, for example, you are trying to locate a particular IDE action that you have forgotten how to trigger.

### Tool windows

Instead of using preset perspectives, Android Studio follows your context and automatically brings up relevant tool windows as you work. By default, the most commonly used tool windows are pinned to the tool window bar at the edges of the application window.

* To expand or collapse a tool window, click the tool’s name in the tool window bar. You can also drag, pin, unpin, attach, and detach tool windows.
* To return to the current default tool window layout, click **Window > Restore Default Layout** or customize your default layout by clicking **Window > Store Current Layout as Default**.
* To show or hide the entire tool window bar, click the window icon https://developer.android.com/studio/images/intro/window-icon_2-1_2x.png in the bottom left-hand corner of the Android Studio window.
* To locate a specific tool window, hover over the window icon and select the tool window from the menu.

You can also use keyboard shortcuts to open tool windows. Table 1 lists the shortcuts for the most common windows.

**Table 1.** Keyboard shortcuts for some useful tool windows.

|  |  |  |
| --- | --- | --- |
| **Tool window** | **Windows and Linux** | **Mac** |
| Project | Alt+1 | Command+1 |
| Version Control | Alt+9 | Command+9 |
| Run | Shift+F10 | Control+R |
| Debug | Shift+F9 | Control+D |
| Logcat | Alt+6 | Command+6 |
| Return to Editor | Esc | Esc |
| Hide All Tool Windows | Control+Shift+F12 | Command+Shift+F12 |

If you want to hide all toolbars, tool windows, and editor tabs, click **View > Enter Distraction Free Mode**. This enables *Distraction Free Mode*. To exit Distraction Free Mode, click **View > Exit Distraction Free Mode**.

You can use *Speed Search* to search and filter within most tool windows in Android Studio. To use Speed Search, select the tool window and then type your search query.

For more tips, see [Keyboard shortcuts](https://developer.android.com/studio/intro/keyboard-shortcuts).

### Code completion

Android Studio has three types of code completion, which you can access using keyboard shortcuts.

**Table 2.** Keyboard shortcuts for code completion.

|  |  |  |  |
| --- | --- | --- | --- |
| **Type** | **Description** | **Windows and Linux** | **Mac** |
| Basic Completion | Displays basic suggestions for variables, types, methods, expressions, and so on. If you call basic completion twice in a row, you see more results, including private members and non-imported static members. | Control+Space | Control+Space |
| Smart Completion | Displays relevant options based on the context. Smart completion is aware of the expected type and data flows. If you call Smart Completion twice in a row, you see more results, including chains. | Control+Shift+Space | Control+Shift+Space |
| Statement Completion | Completes the current statement for you, adding missing parentheses,etc. | Control+Shift+Enter | Shift+Command+Enter |

You can also perform quick fixes and show intention actions by pressing **Alt+Enter**.

### Find sample code

The Code Sample Browser in Android Studio helps you find high-quality, Google-provided Android code samples based on the currently highlighted symbol in your project. For more information, see [Find sample code](https://developer.android.com/studio/write/sample-code).

### Navigation

Here are some tips to help you move around Android Studio.

* Switch between your recently accessed files using the *Recent Files* action. Press **Control+E** (**Command+E** on a Mac) to bring up the Recent Files action. By default, the last accessed file is selected. You can also access any tool window through the left column in this action.
* View the structure of the current file using the *File Structure* action. Bring up the File Structure action by pressing **Control+F12** (**Command+F12** on a Mac). Using this action, you can quickly navigate to any part of your current file.
* Search for and navigate to a specific class in your project using the *Navigate to Class* action. Bring up the action by pressing **Control+N** (**Command+O** on a Mac). Navigate to Class supports sophisticated expressions, including camel humps, paths, line navigate to, middle name matching, and many more. If you call it twice in a row, it shows you the results out of the project classes.
* Navigate to a file or folder using the *Navigate to File* action. Bring up the Navigate to File action by pressing **Control+Shift+N** (**Command+Shift+O** on a Mac). To search for folders rather than files, add a / at the end of your expression.
* Navigate to a method or field by name using the *Navigate to Symbol* action. Bring up the Navigate to Symbol action by pressing **Control+Shift+Alt+N** (**Command+Option+O** on a Mac).
* Find all the pieces of code referencing the class, method, field, parameter, or statement at the current cursor position by pressing **Alt+F7** (**Option+F7** on a Mac).

### Style and formatting

As you edit, Android Studio automatically applies formatting and styles as specified in your code style settings. You can customize the code style settings by programming language, including specifying conventions for tabs and indents, spaces, wrapping and braces, and blank lines. To customize your code style settings, click **File > Settings > Editor > Code Style** (**Android Studio > Preferences > Editor > Code Style** on a Mac.)

Although the IDE automatically applies formatting as you work, you can also explicitly call the *Reformat Code* action by pressing **Control+Alt+L** (**Opt+Command+L** on a Mac), or auto-indent all lines by pressing **Control+Alt+I** (**Control+Option+I** on a Mac).

### Version control basics

Android Studio supports a variety of version control systems (VCS’s), including Git, GitHub, CVS, Mercurial, Subversion, and Google Cloud Source Repositories.

After importing your app into Android Studio, use the Android Studio VCS menu options to enable VCS support for the desired version control system, create a repository, import the new files into version control, and perform other version control operations:

1. From the Android Studio **VCS** menu, click **Enable Version Control Integration**.
2. From the drop-down menu, select a version control system to associate with the project root, and then click **OK**.

The VCS menu now displays a number of version control options based on the system you selected.

**Note:** You can also use the **File > Settings > Version Control** menu option to set up and modify the version control settings.

## Gradle build system

Android Studio uses Gradle as the foundation of the build system, with more Android-specific capabilities provided by the [Android plugin for Gradle](https://developer.android.com/studio/releases/gradle-plugin). This build system runs as an integrated tool from the Android Studio menu, and independently from the command line. You can use the features of the build system to do the following:

* Customize, configure, and extend the build process.
* Create multiple APKs for your app, with different features using the same project and modules.
* Reuse code and resources across sourcesets.

By employing the flexibility of Gradle, you can achieve all of this without modifying your app's core source files. Android Studio build files are named build.gradle. They are plain text files that use [Groovy](http://groovy-lang.org/) syntax to configure the build with elements provided by the Android plugin for Gradle. Each project has one top-level build file for the entire project and separate module-level build files for each module. When you import an existing project, Android Studio automatically generates the necessary build files.

To learn more about the build system and how to configure, see [Configure your build](https://developer.android.com/studio/build).

### Build variants

The build system can help you create different versions of the same application from a single project. This is useful when you have both a free version and a paid version of your app, or if you want to distribute multiple APKs for different device configurations on Google Play.

For more information about configuring build variants, see [Configure build variants](https://developer.android.com/studio/build/build-variants).

### Multiple APK support

Multiple APK support allows you to efficiently create multiple APKs based on screen density or ABI. For example, you can create separate APKs of an app for the hdpi and mdpi screen densities, while still considering them a single variant and allowing them to share test APK, javac, dx, and ProGuard settings.

For more information about multiple APK support, read [Build multiple APKs](https://developer.android.com/studio/build/configure-apk-splits).

### Resource shrinking

Resource shrinking in Android Studio automatically removes unused resources from your packaged app and library dependencies. For example, if your application is using [Google Play services](https://developers.google.com/android/guides/overview) to access Google Drive functionality, and you are not currently using [Google Sign-In](https://developer.android.com/training/sign-in), then resource shrinking can remove the various drawable assets for theSignInButton buttons.

**Note:** Resource shrinking works in conjunction with code shrinking tools, such as ProGuard.

For more information on shrinking code and resources, see [Shrink your code and resources](https://developer.android.com/studio/build/shrink-code).

### Managing dependencies

Dependencies for your project are specified by name in the build.gradle file. Gradle takes care of finding your dependencies and making them available in your build. You can declare module dependencies, remote binary dependencies, and local binary dependencies in your build.gradle file. Android Studio configures projects to use the Maven Central Repository by default. (This configuration is included in the top-level build file for the project.) For more information about configuring dependencies, read [Add build dependencies](https://developer.android.com/studio/build/dependencies).

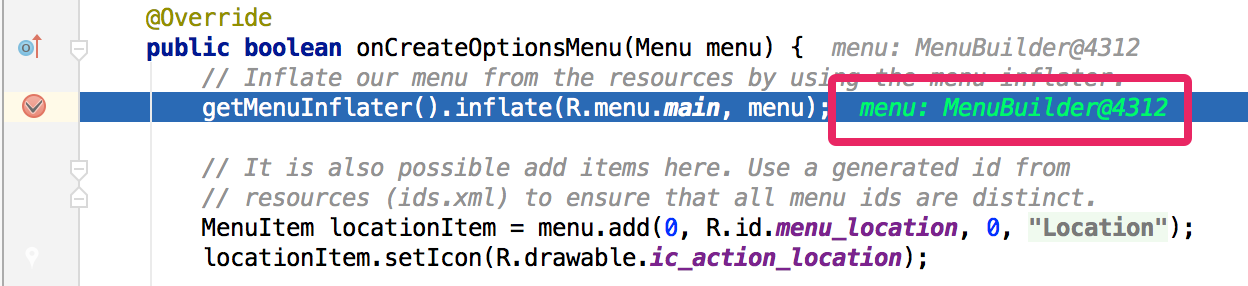
## Debug and profile tools

Android Studio assists you in debugging and improving the performance of your code, including inline debugging and performance analysis tools.

### Inline debugging

Use inline debugging to enhance your code walk-throughs in the debugger view with inline verification of references, expressions, and variable values. Inline debug information includes:

* Inline variable values
* Referring objects that reference a selected object
* Method return values
* Lambda and operator expressions
* Tooltip values



*3.3* *AN INLINE VARIABLE VALUE.*

To enable inline debugging, in the **Debug** window, click **Settings** https://developer.android.com/images/tools/studio-debug-settings-icon.pngand select the checkbox for **Show Values Inline**.

### Performance profilers

Android Studio provides performance profilers so you can more easily track your app’s memory and CPU usage, find deallocated objects, locate memory leaks, optimize graphics performance, and analyze network requests. With your app running on a device or emulator, open the **Android Profiler** tab.

For more information about performance profilers, see [Performance profiling tools](https://developer.android.com/studio/profile).

### Heap dump

When you’re profiling memory usage in Android Studio, you can simultaneously initiate garbage collection and dump the Java heap to a heap snapshot in an Android-specific HPROF binary format file. The HPROF viewer displays classes, instances of each class, and a reference tree to help you track memory usage and find memory leaks.

For more information about working with heap dumps, see [Inspect the heap and allocations](https://developer.android.com/studio/profile/memory-profiler).

### Memory Profiler

You can use Memory Profiler to track memory allocation and watch where objects are being allocated when you perform certain actions. Knowing these allocations enables you to optimize your app’s performance and memory use by adjusting the method calls related to those actions.

For information about tracking and analyzing allocations, see [Inspect the heap and allocations](https://developer.android.com/studio/profile/memory-profiler).

### Data file access

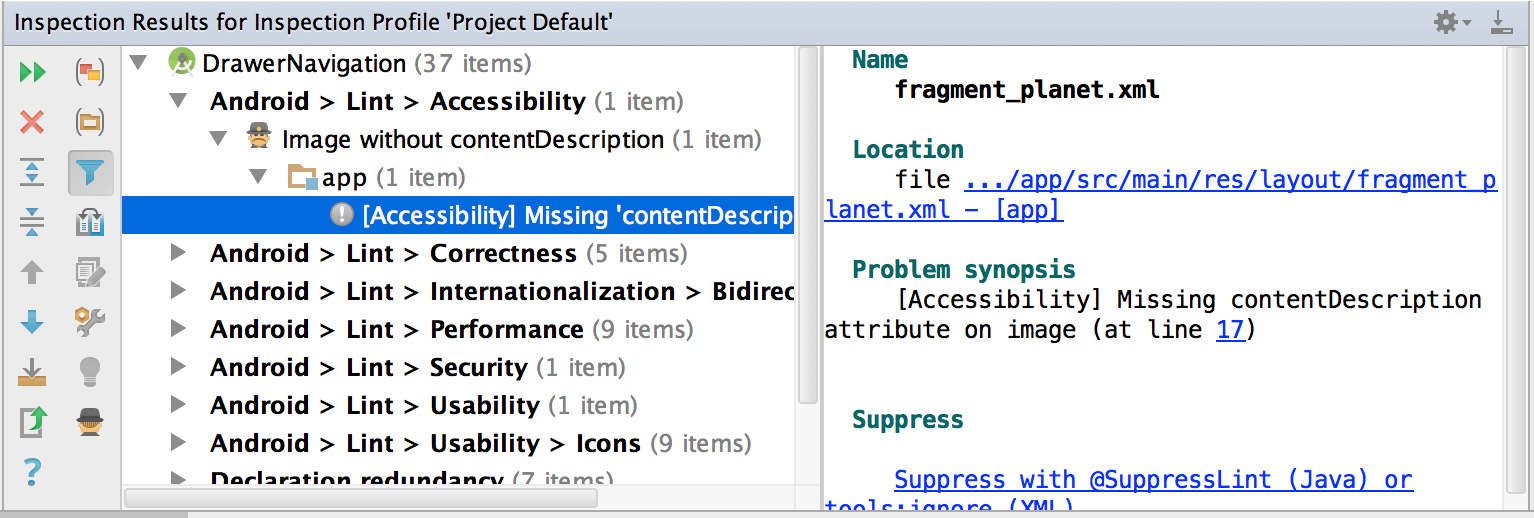
The Android SDK tools, such as [Systrace](https://developer.android.com/topic/performance/tracing/command-line), and [logcat](https://developer.android.com/studio/debug/am-logcat), generate performance and debugging data for detailed app analysis.

To view the available generated data files, open the Captures tool window. In the list of the generated files, double-click a file to view the data. Right-click any .hprof files to convert them to the standard [Investigate your RAM usage](https://developer.android.com/studio/profile/investigate-ram) file format.

### Code inspections

Whenever you compile your program, Android Studio automatically runs configured [Lint](https://developer.android.com/studio/write/lint) and other [IDE inspections](https://www.jetbrains.com/help/idea/2020.1/code-inspection.html) to help you easily identify and correct problems with the structural quality of your code.

The Lint tool checks your Android project source files for potential bugs and optimization improvements for correctness, security, performance, usability, accessibility, and internationalization.



*3.4 THE RESULTS OF A LINT INSPECTION IN ANDROID STUDIO.*

In addition to Lint checks, Android Studio also performs IntelliJ code inspections and validates annotations to streamline your coding workflow.

For more information, see [Improve your code with lint checks](https://developer.android.com/studio/write/lint).

### Annotations in Android Studio

Android Studio supports annotations for variables, parameters, and return values to help you catch bugs, such as null pointer exceptions and resource type conflicts. The Android SDK Manager packages the Support-Annotations library in the Android Support Repository for use with Android Studio. Android Studio validates the configured annotations during code inspection.

For more details about Android annotations, see [Improve code inspection with annotations](https://developer.android.com/studio/write/annotations).

### Log messages

When you build and run your app with Android Studio, you can view [adb](https://developer.android.com/studio/command-line/adb) output and device log messages in the [**Logcat** window](https://developer.android.com/studio/debug/am-logcat).

### Performance profiling

If you want to profile your app's CPU, memory, and network performance, open the [Android Profiler](https://developer.android.com/studio/profile/android-profiler), by clicking **View > Tool Windows > Android Profiler**.

## Sign in to your developer account

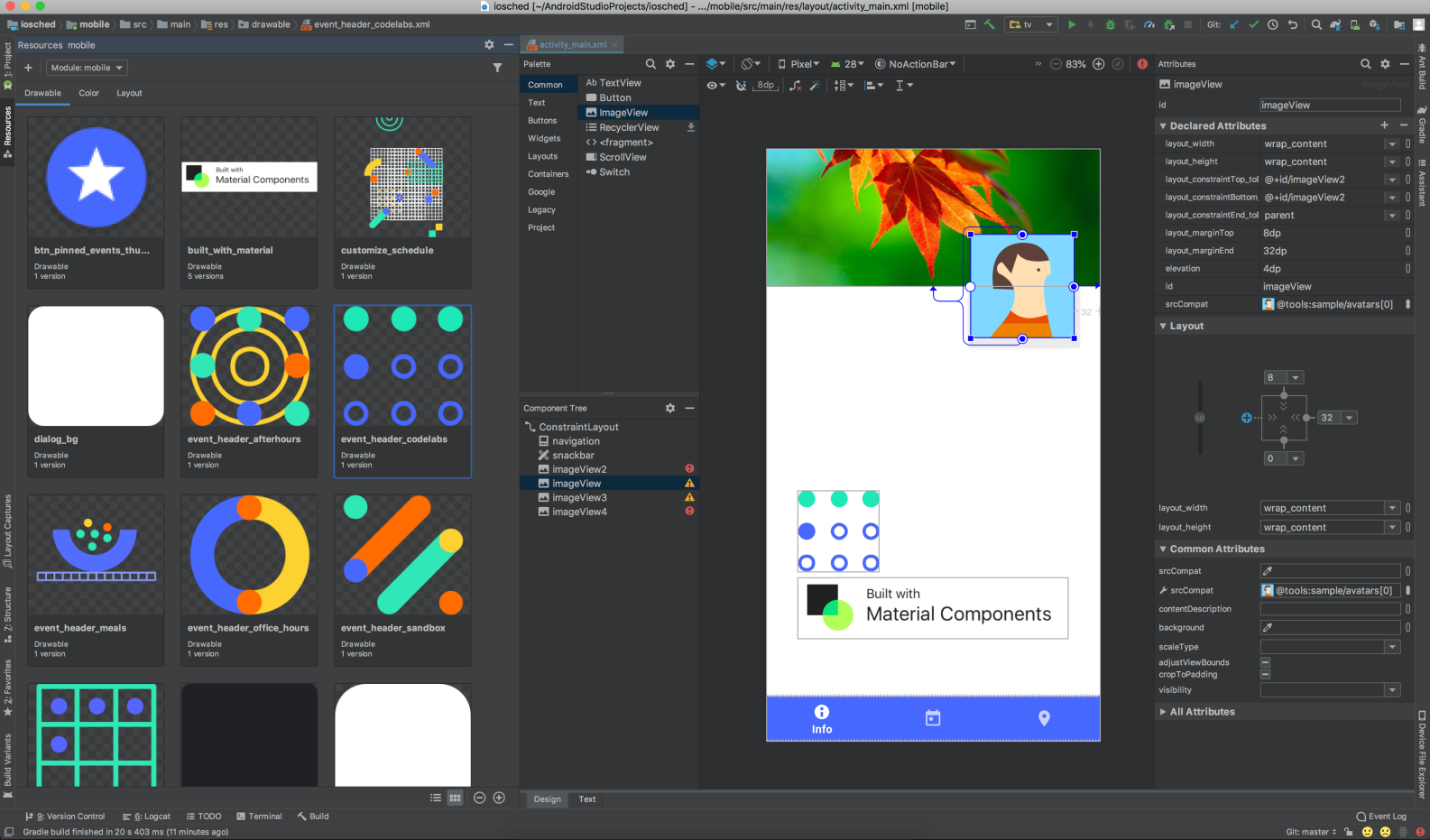
You can sign in to your developer account in Android Studio to access additional tools that requires authentication, such as [Cloud Tools for Android Studio](https://cloud.google.com/tools/android-studio/docs/) and the [App Actions test tool](https://developers.google.com/assistant/app/test-tool). By signing in, you give those tools permission to view and manage your data across Google services.

After you open a project in Android Studio, you can sign in to your developer account or switch developer accounts, as follows:

1. Click the profile icon https://developer.android.com/studio/images/intro/profile-icon.png at the end of the toolbar.

Click the profile icon at the end of the toolbar to sign in.

1. In the window that appears, do one of the following:
   * If you're not yet signed in, click **Sign In** and allow Android Studio to access the listed services.
   * If you're already signed in, click **Add Account** to sign in with another Google account. Alternatively, you can click **Sign Out** and repeat the previous steps to sign in to a different account.



*3.5 ANDROID STUDIO*

# CHAPTER 8 : REFERENCES

1. **Detection of nitrogen, phosphorus, and potassium (NPK) nutrients of soil using optical transducer,** MarianahMasrie, Mohamad SyamimAizuddin Rosman, Rosidah Sam, ZuriatiJanin, 2017 IEEE 4th International Conference on Smart Instrumentation, Measurement and Application (ICSIMA), 1-4, 2017.
2. **The tracking of a moving object by a mobile robot following the object’s sound,** Jongho Han, Sunsin Han, Jangmyung Lee, Journal of intelligent & robotic systems 71 (1), 31-42, 2013.
3. **Smart irrigation system using cloud and Internet of Things,** Suresh Koduru, VGD Prasad Reddy Padala, Preethi Padala, Proceedings of 2nd international conference on communication, computing and networking, 195-203, 2019.
4. **AUTONOMOUS NAVIGATION WITH ROS FOR A MOBILE ROBOT IN AGRICULTURAL FIELDS,** Mark A. Post, Alessandro Bianco and Xiu T. Yan, 2016.
5. **AERIAL FARM ROBOT SYSTEM FOR CROP DUSTING, PLANTING, FERTILIZING AND OTHER FIELD JOBS,** HarmBUREMA and Anatoly FILIN, 2014.
6. **FLEXIBLE AGRICULTURAL AUTOMATION,**Brian P. Hanley, 2002.
7. **AGRICULTURAL ROBOT SYSTEM AND METHOD,** Harvey Koselka andBret Wallach, 2006.
8. **SMART IRRIGATION WITH FIELD PROTECTION AND CROP HEALTH MONITORING SYSTEM USING AUTONOMOUS ROVER,** S. Gobhinath, M. Devi Darshini, K. Durga, R. Hari Priyanga,2019.
9. **SIMPLIFIED MODELLING AND BACKSTEPPING CONTROL OF THE LONG ARM AGRICULTURAL ROVER,** NapasoolWongvanich, SungwanBoksuwanandAbdulhafizChesof, 2020.