

**ADAMA SCIENCE AND TECHNOLOGY UNIVERSITY**

**SCHOOL OF ELECTRICAL ENGINEERING AND COMPUTING**

**PROGRAM OF COMPUTER SCIENCE AND ENGINEERING**

**A SENIOR PROJECT DOCUMENTATION SUBMITTED FOR THE PARTIAL FULFILLMENT OF DEGREE PROGRAM AT DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**PROJECT TITLE: - TEFF SOWING AGROBOT (TSA)**

|  |  |  |
| --- | --- | --- |
|  | **GROUP MEMBERS** |  |
| Name | Id | Group |
| 1) Mengistu Abebe | R/00688/07 | II |
| 2) Merga W/yohannes | R/00691/07 | II |
| 3) Tizita Daba | R/00958/07 | II |
| 4) Getu Lakew | R/00466/07 | I |
| 5) Abdin Tekalign | R/00009/07 | I |

Submission date: July 23, 2019

Supervised by: Yordanos Gebeyehu

# Declaration

|  |  |  |  |
| --- | --- | --- | --- |
| **Submitted by** |  |  |  |
| Mengistu Abebe |  | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | July 23, 2019 |
| Student |  | Signature | Date |
| Merga W/yohannes | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | | July 23, 2019 |
| Student |  | Signature | Date |
| Getu Lakew | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | | July 23, 2019 |
| Student |  | Signature | Date |
| Tizita Daba |  | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | July 23, 2019 |
| Student |  | Signature | Date |
| Abdin Tekalign |  | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | July 23, 2019 |
| Student |  | Signature | Date |
| **Approved by** |  |  |  |
| 1. Mrs. Yordanos | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | | July 23, 2019 |
| Advisor |  | Signature | Date |
| 2. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| Chairman, Dept.’s |  | Signature | Date |

Senior project Committee

Department

|  |  |  |  |
| --- | --- | --- | --- |
| 3. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| Head of the Dept. |  | Signature | Date |
|  |  |  |  |

# Acknowledgement

The success and final outcome of this project required a lot of guidance and assistance from many people and we are extremely privileged to have got this all along the completion of our project. All that we have done is only due to such supervision and assistance and we would not forget to thank them.

We would like to express our deepest gratitude to our advisor, instructor Mrs. Yordanos Gebeyehu, for her outstanding leadership, support, encouragement, and guidance. Throughout this thesis project, she has been a great and effective advisor and has offered all possible assistance to improve the quality of our work.

Our sincere thanks will go to Mr. Fitsum, Agricultural mechanization manager and Mr. Yonas, Agricultural mechanization vice manager in Awash Agricultural Institute for their encouragement and more over for their timely support and guidance on giving necessary information in completing the project.

# Table of Contents

[Declaration i](#_Toc14711084)

[Acknowledgement ii](#_Toc14711085)

[Table of Contents iii](#_Toc14711086)

[List of Figure vi](#_Toc14711087)

[List of Tables viii](#_Toc14711088)

[List of Acronyms viii](#_Toc14711089)

[Abstract xi](#_Toc14711090)

[CHAPTER ONE 1](#_Toc14711091)

[INTRODUCTION OF THE PROJECT PROCESS 1](#_Toc14711092)

[**1.1** **Introduction** 1](#_Toc14711093)

[**1.2** **Background information** 2](#_Toc14711094)

[**1.3** **Statement of the problem** 3](#_Toc14711095)

[**1.4** **Purpose of the project** 4](#_Toc14711096)

[**1.5** **Objective** 4](#_Toc14711097)

[**1.5.1** General objective 4](#_Toc14711098)

[**1.5.2** Specific objective 4](#_Toc14711099)

[**1.6** **Feasibility Study** 5](#_Toc14711100)

[**1.6.1** Operational Feasibility 5](#_Toc14711101)

[**1.6.2** Technical Feasibility 6](#_Toc14711102)

[**1.6.3** Economical Feasibility 6](#_Toc14711103)

[**1.7** **Scope and limitation** 7](#_Toc14711104)

[**1.7.1** **Scope** 7](#_Toc14711105)

[**1.7.2** **Limitation** 7](#_Toc14711106)

[**1.8** **Significance of the project** 7](#_Toc14711107)

[**1.9** **Methodology** 8](#_Toc14711108)

[**1.9.1** Fact finding methodology 8](#_Toc14711109)

[**1.9.2** System Development Methodology 9](#_Toc14711110)

[**1.9.3** Development Tools 9](#_Toc14711111)

[**1.10** **Constraints and Assumptions** 11](#_Toc14711112)

[**1.10.1** Constraints 11](#_Toc14711113)

[**1.10.2** Assumptions 11](#_Toc14711114)

[**1.11** **Testing plan** 11](#_Toc14711115)

[**1.11.1** Unit test 11](#_Toc14711116)

[**1.11.2** Integration test 11](#_Toc14711117)

[**1.12** **Overview of project phase** 12](#_Toc14711118)

[**1.13** **Schedule** 13](#_Toc14711119)

[**1.14** **Team Composition** 13](#_Toc14711120)

[CHAPTER TWO 15](#_Toc14711121)

[2 CURRENT SYSTEM 15](#_Toc14711122)

[**2.1** **Major Functionality of the Current System** 15](#_Toc14711123)

[**2.2** **User of the Current System** 15](#_Toc14711124)

[**2.3** **Problem of the Current System** 16](#_Toc14711125)

[**2.4** **Business role of the systems** 17](#_Toc14711126)

[CHAPTER THREE 18](#_Toc14711127)

[3 PROPOSED SYSTEM 18](#_Toc14711128)

[**3.1** **Overview** 18](#_Toc14711129)

[**3.2** **Functional Requirement** 18](#_Toc14711130)

[**3.3** **Non–Functional Requirement** 19](#_Toc14711131)

[**3.4** **System Model** 21](#_Toc14711132)

[**3.4.1** **Scenario** 21](#_Toc14711133)

[**3.5** **Use case Model** 26](#_Toc14711134)

[**3.5.1** Actor Identification 26](#_Toc14711135)

[**3.5.2** Use case diagram and Description 27](#_Toc14711136)

[**3.6** **Object Model** 44](#_Toc14711137)

[**3.7** **Class Diagram** 46](#_Toc14711138)

[**3.8** **Dynamic model** 46](#_Toc14711139)

[**3.8.1** Sequence Diagram 46](#_Toc14711140)

[**3.9** **Activity Diagram** 54](#_Toc14711141)

[**3.10** **State Diagram** 61](#_Toc14711142)

[CHAPTER FOUR 67](#_Toc14711143)

[4 SYSTEM DESIGN 67](#_Toc14711144)

[4.1 Overview 68](#_Toc14711145)

[**4.1.1** Purpose of the System Design 68](#_Toc14711146)

[**4.1.2** Design Goal 68](#_Toc14711147)

[**4.2** **Proposed system** 70](#_Toc14711148)

[**4.2.1** **Overview** 70](#_Toc14711149)

[**4.2.2** **System process** 71](#_Toc14711150)

[**4.2.3** **Subsystem decomposition** 73](#_Toc14711151)

[**4.2.4** **Hardware/software mapping** 76](#_Toc14711152)

[**4.2.5** **Persistent data management** 77](#_Toc14711153)

[**4.2.6** **Component diagram** 80](#_Toc14711154)

[**4.2.7** **Deployment diagram** 81](#_Toc14711155)

[**4.2.8** **Database design** 82](#_Toc14711156)

[**4.2.9** **Access control** 82](#_Toc14711157)

[**4.2.10** **User interface design** 83](#_Toc14711158)

[Chapter Five 85](#_Toc14711159)

[5 Implementation 85](#_Toc14711160)

[5.1 Overview 85](#_Toc14711161)

[5.2 Tools and technologies utilized during system development 85](#_Toc14711162)

[5.3 Prototype Setup 105](#_Toc14711163)

[5.4 Implementation Detail 109](#_Toc14711164)

[CHAPTER SIX 110](#_Toc14711165)

[6 SYSTEM TESTING 110](#_Toc14711166)

[6.1 Introduction 110](#_Toc14711167)

[6.2 Scope 110](#_Toc14711168)

[6.3 Resources 110](#_Toc14711169)

[6.4 Schedule 110](#_Toc14711170)

[6.5 Features to be tested and not to be tested 111](#_Toc14711171)

[**6.5.1** Features to be tested 111](#_Toc14711172)

[**6.5.2** Features not to be tested 111](#_Toc14711173)

[6.6 Pass/fail criteria 111](#_Toc14711174)

[6.7 Test case specification 112](#_Toc14711175)

[6.8 Estimated risk and contingency plan 114](#_Toc14711176)

[CHAPTER SEVEN 115](#_Toc14711177)

[7 Conclusion and Recommendation 115](#_Toc14711178)

[7.1 Conclusion 115](#_Toc14711179)

[7.2 Recommendation 115](#_Toc14711180)

[Annex 116](#_Toc14711181)

[Prototype development 129](#_Toc14711182)

[References 135](#_Toc14711183)

# List of Figure

[Figure 1 Use Case diagram 26](#_Toc14773737)

[Figure 2 Class diagram for Teff seed row sowing Agrobot 45](#_Toc14773738)

[Figure 3 sequence diagram Agrobot control movement 46](#_Toc14773739)

[Figure 4 sequence diagram User control movement 47](#_Toc14773740)

[Figure 5 sequence diagram User control speed 47](#_Toc14773741)

[Figure 6 sequence diagram Agrobot determine critical seed level 48](#_Toc14773742)

[Figure 7 sequence diagram user enters input distance to sow 48](#_Toc14773743)

[Figure 8 sequence diagram Agrobot determine seed level 49](#_Toc14773744)

[Figure 9 sequence diagram user login to Agrobot 49](#_Toc14773745)

[Figure 10 sequence diagram user logout 50](#_Toc14773746)

[Figure 11 sequence diagram user generate report 50](#_Toc14773747)

[Figure 12 sequence diagram user can view details 51](#_Toc14773748)

[Figure 13 sequence diagram for registration 51](#_Toc14773749)

[Figure 14 sequence diagram for Agrobot control input distance 52](#_Toc14773750)

[Figure 16 Sequence diagram for agrobot can sow 52](#_Toc14773751)

[Figure 17 activity diagram Agrobot control movement 53](#_Toc14773752)

[Figure 18 activity diagram Agrobot determine critical seed level 53](#_Toc14773753)

[Figure 19 activity diagram Agrobot determine input distance to sow 54](#_Toc14773754)

[Figure 20 activity diagram Agrobot determine level of seed 54](#_Toc14773755)

[Figure 21 activity diagram user login to Agrobot 55](#_Toc14773756)

[Figure 22 activity diagram user generate report 55](#_Toc14773757)

[Figure 23 activity diagram user control movement 56](#_Toc14773758)

[Figure 24 activity diagram user control speed 56](#_Toc14773759)

[Figure 25 activity diagram user input distance to sow 57](#_Toc14773760)

[Figure 26 activity diagram user can view details 57](#_Toc14773761)

[Figure 27 activity diagram for registration 58](#_Toc14773762)

[Figure 28 activity diagram for User can logout 58](#_Toc14773763)

[Figure 30 activity diagram for agrobot can sow 59](#_Toc14773764)

[Figure 31 state chart diagram Agrobot control movement 60](#_Toc14773765)

[Figure 32 state chart diagram user control movement 60](#_Toc14773766)

[Figure 33 state chart diagram user control Speed 61](#_Toc14773767)

[Figure 34 state chart diagram Agrobot determine the critical seed level 61](#_Toc14773768)

[Figure 35 state chart diagram Agrobot determine seed level 62](#_Toc14773769)

[Figure 36 state chart diagram user generate report 62](#_Toc14773770)

[Figure 37 state chart diagram user enters input distance to sow 63](#_Toc14773771)

[Figure 38 state chart diagram user login to Agrobot 63](#_Toc14773772)

[Figure 39 state chart diagram user can view details 64](#_Toc14773773)

[Figure 40 state chart diagram of registration 64](#_Toc14773774)

[Figure 41 state chart diagram Agrobot input distance to sow 65](#_Toc14773775)

[Figure 42 state chart diagram for User can logout 65](#_Toc14773776)

[Figure 41 State chart diagram for User can sow 66](#_Toc14773777)

[Figure 45 system architecture of TSA 70](#_Toc14773778)

[Figure 46 Agrobot system process TSA 70](#_Toc14773779)

[Figure 47 Agrobot determine system overview 71](#_Toc14773780)

[Figure 48 Agrobot control and manage input from user system overview 71](#_Toc14773781)

[Figure 49 Agrobot software system process of TSA 72](#_Toc14773782)

[Figure 50 Agrobot software side system over view 73](#_Toc14773783)

[Figure 51 subsystem decomposition of TSA level 1 74](#_Toc14773784)

[Figure 52 subsystem decomposition of TSA level 2 75](#_Toc14773785)

[Figure 53 hardware/software mapping of TSA for Agrobot software side 77](#_Toc14773786)

[Figure 54 hardware/software mapping of TSA for Agrobot 77](#_Toc14773787)

[Figure 55 object for user 78](#_Toc14773788)

[Figure 56 object for Agrobot 79](#_Toc14773789)

[Figure 57 object for farmland covered 79](#_Toc14773790)

[Figure 58 object for report 79](#_Toc14773791)

[Figure 59 object for sensor value 80](#_Toc14773792)

[Figure 60 object for reset password 80](#_Toc14773793)

[Figure 61 Component diagram of TSA 80](#_Toc14773794)

[Figure 62 deployment diagram of TSA for Agrobot software side 81](#_Toc14773795)

[Figure 63 deployment diagram of TSA for Agrobot side 81](#_Toc14773796)

[Figure 64 database design of TSA 82](#_Toc14773797)

[Figure 65 user interface design of TSA for Agrobot software side 84](#_Toc14773798)

[Figure 66 user interface design of TSA for Agrobot side 85](#_Toc14773799)

[Figure 67 System block diagram 86](#_Toc14773800)

[Figure 68 Arduino-Uno and pin configurations 87](#_Toc14773801)

[Figure 69 Stepper motor 91](#_Toc14773802)

[Figure 70 Interior of stepper motor 92](#_Toc14773803)

[Figure 71 16×2 **LCD** 92](#_Toc14773804)

[Figure 72 Keypad 94](#_Toc14773805)

[Figure 73 Breadboard 95](#_Toc14773806)

[Figure 74 Ultrasonic sensor 96](#_Toc14773807)

[Figure 75 **ESP NodeMCU** 97](#_Toc14773808)

[Figure 76 Push Button 99](#_Toc14773809)

[Figure 77 IR Sensor 100](#_Toc14773810)

[Figure 78 Resistors 100](#_Toc14773811)

[Figure 79 AgrobotPrototype Full Designs 101](#_Toc14773812)

[Figure 80 Measurements of the wood frame 102](#_Toc14773813)

[Figure 81 Seed dispenser 103](#_Toc14773814)

[Figure 82 Full designs of seed container and dispenser 103](#_Toc14773815)

[Figure 83 Ball bearing 103](#_Toc14773816)

[Figure 84 Plastic tube 104](#_Toc14773817)

[Figure 85 Wheel 104](#_Toc14773818)

[Figure 86 Belt 105](#_Toc14773819)

[Figure 87 Block diagram for agrobot control movement 107](#_Toc14773820)

[Figure 88 Block diagram for agrobot manage input 107](#_Toc14773821)

[Figure 89 Block diagram app control speed 108](#_Toc14773822)

[Figure 90 Block diagram app manage input distance 108](#_Toc14773823)

[Figure 91 Block diagram app control movement 109](#_Toc14773824)

[Figure 92 Block diagram critical seed notification 110](#_Toc14773825)

[Figure 93 Block diagram determine seed level 110](#_Toc14773826)

[Figure 94 Block diagram of view details 111](#_Toc14773827)

[Figure *95* Determine the seed level and the position of agrobot on Proteus 130](#_Toc14773828)

[Figure 96 when system is off 131](#_Toc14773829)

[Figure *97* user inserts Input Distance to sow and display on the lcd on Proteus 131](#_Toc14773830)

[Figure 98 Control movement 132](#_Toc14773831)

[Figure 99 Manage input distance and row spacing Interface for English 133](#_Toc14773832)

[Figure 100 Manage input distance and row spacing Interface for Amharic 134](#_Toc14773833)

[Figure 101 Manage input distance and row spacing Interface for Afaan Oromoo 135](#_Toc14773834)

# List of Tables

[Table 1 Average cost analysis of Agrobot 6](#_Toc5223547)

[Table 2 Client and server-side tools 9](#_Toc5223548)

[Table 3 Other tools 9](#_Toc5223549)

[Table 4 Schedule 13](#_Toc5223550)

[Table 5 Team composition 14](#_Toc5223551)

[Table 6 use case description of login 28](#_Toc5223552)

[Table 7 use case description of Notify critical Seed 29](#_Toc5223553)

[Table 8 use case description of row spacing 30](#_Toc5223554)

[Table 9 use case description of farmland covered 31](#_Toc5223555)

[Table 10 use case description of level of seed 32](#_Toc5223556)

[Table 11 use case description of take input distance 33](#_Toc5223557)

[Table 12 use case description of control speed 34](#_Toc5223558)

[Table 13 use case description of control movement 35](#_Toc5223559)

[Table 14 use case description of control movement 36](#_Toc5223560)

[Table 15 use case description of generate report 37](#_Toc5223561)

[Table 16 Table use case description of view position 38](#_Toc5223562)

[Table 17 Table use case description of view farmland covered 39](#_Toc5223563)

[Table 18 use case description of view level of seed 40](#_Toc5223564)

[Table 19 use case description of view critical seed level 41](#_Toc5223565)

[Table 20 use case description of logout 42](#_Toc5223566)

[Table 21 use case description of registration 42](#_Toc5223567)

[Table 22 Object model 44](#_Toc5223568)

[Table 23 subsystem description 75](#_Toc5223569)

[Table 24 access control 82](#_Toc5223570)

# List of Acronyms

A

Agrobot

Agricultural Robot ·

ATA

Agricultural Transformation Agency ·

I

IDE

Integrated Development Environment ·

IR

infrared ·

M

Microc

ANSI C compiler for PIC devices ·

MoA

Ministry of Agriculture ·

MS-ppt

Microsoft power point ·

MS-word

Microsoft word ·

MYSQL

My Structured Query Language ·

P

PHP

Hypertext Pre Processor ·

R

RAM

Random Access Memory ·

S

SQLite

relational database management system ·

SRS

Software Requirement Specification ·

T

TSA

Teff Sowing Agrobot ·

U

UML

Unified Modelling Language ·

Abstract  
Seeding Agrobot is an automation [robot](https://searchenterpriseai.techtarget.com/definition/robot) used in farming to sow seed in row manner. Teff is a highly valuable grain for Ethiopian people both in production amount and in consumption level. So, the agricultural system in Ethiopia should be advanced to reduce the efforts of farmers. Currently Teff production system used by the majority of farmer is very backward and traditional, most of farmers in the country broadcast Teff seeds. Even though row planting activity has yield increasing capability it has difficulties with implementation like time consumption, increased labor force, tediousness of the job. This paper contains starting from data gathering, system analysis, and system design of Teff sowing Agrobot. This project intends to apply the achievements made by modern technologies in the improvement of Teff yield and farmers’ level of perception towards the application of modern Teff production. The Agrobot that we are going to develop will help Ethiopian farmers to exercise Teff row planting which enhance productivity of farmer with the same given input and farm land. The robot is small and electrically operated so that there is no need of mechanical support from human or animal. It provides row spacing, speed controlling, coverage calculation mechanism. It also solves the challenges that farmers faced with the application of row planting of Teff seed. The Agrobot puts the seeds at desired row and its main objective is to increase productivity, minimize the work of farmer and also reduces the time for seed sowing operation.

Keywords: Teff, Row planting, Designing and Prototyping, Testing, Agrobot.

# CHAPTER ONE

# INTRODUCTION OF THE PROJECT PROCESS

## **Introduction**

Seeding Agrobot is an automation [robot](https://searchenterpriseai.techtarget.com/definition/robot) used in farming to sow seed in row manner and helps to improve efficiency and reduce reliance on manual labor by automating manual row seeding mechanism [1]. Future farms are expected to be sown by automated robots called sowing Agrobot that will sow seed keeping the desired row spacing with controlled speed, all the while collecting valuable data [2].

In current world, every process is getting automated and people are getting used to adopt Smart techniques to get their work done. It can be seen that with flow of time, how seed Sowing techniques and equipment’s have kept on progressing. Proper seed sowing is very Important part of agricultural process and for the same purpose hand operated seed sowing Machine have been designed and developed [3]. Despite agriculture being one of the most Important fields for determining the growth of a country, it is lagging in terms of smart working. If technology is introduced in farming techniques there are chances that ever-growing populating in the coming future might be fed adequately. To suffice such a large amount, agricultural yield must also be increased rapidly [3].

Automation in seed sowing will help in proper use of available resources. To implement automation in the process of sowing seeds in agricultural Farming, the machines that are already being used can be improved in design or new machines can be developed to do the necessary operations. But these machines be cost effective and be affordable to the farmers. This will help to increase output with same amount of input by sowing the seed at proper distance row wisely so that each seed gives best output as it is known that sowing of seed with proper gap is an important parameter in farming [3].

For an agriculture sector to be successful one needs to add the booming technologies as Input and take care of the processes and at the same time knowing the behavior of the technology and the major role that it is going to play in the sector of one’s interest. In the present growing aspect, the need to utilize the available technologies has become necessity in order to gain the best result [3]. With all these information and thoughts, automation of seed sowing by using Agrobot would reduce the load of farmers and increase the productivity.

The system we are going to develop is an agricultural robot that is used for sowing *Teff* in row wise with its controlling software. This will solve problem arising from manual row planting like tediousness, time consumption, labor force and effectiveness.

## **Background information**

*Teff* is a highly valuable grain for Ethiopian people both in production amount and in consumption level. It is a staple food and a source for more than 15% of calories intake by the total population of the country. More than 6 million households’ life depend on the production of *teff* covering the largest agricultural area of the country than any other types of grain, however the amount of production is not as much as its production coverage and value [4].

*Teff* production system used by the majority of farmers is very backward and traditional, most of the farmers in the country broadcast *Teff* seeds, i.e. scattering seed by hand, at high seed rates. This impedes *Teff* yields because of high amount and uneven distribution of the seeds makes weeding difficult and increased competition with weeds and other *Teff* plants lowers nutrient uptake by the individual *Teff* plant [4][5]. This result in the reduction of *Teff* yield at the harvesting period. In response to this currently there are some works made in partnerships of MoA and ATA farmers are being introduced to modern technologies of row planting of *Teff* with reduced seed rate.[6]

Currently, agricultural extension activities have been concerned with the promotion, adoption and scaling up of row planting practices; and adoption of the practice is seen as the factor for yield enhancement in the country. As a result, manual and machinery-based row planting has become one of the agronomic practices of smallholder farmers in the country [7]. Even though this activity has yield increasing capability it has difficulties with implementation like time consumption, increased labor force, tediousness of the job. Due to these limitations the conventional planting method that is broadcasting seed by hand at high seed rates is exercised by many farmers in our country. But this method reduce yield because uneven distribution of the seeds makes hand weeding and hoeing difficult, and plant competition with weeds lowers growth. This causes yield reduction. However, row planting with proper distance between rows and plant density allows for sufficient aeration, moisture, sunlight and nutrient availability leading to proper root system development.

In 2015, the Nuru Ethiopia Agriculture Program is working with farmers to enhance productivity. Nuru promotes proper spacing and row planting compared to broadcasting seed. This is because farmers have limited land-holdings. When seed is broadcast, it may be too dense or too disperse, resulting in loss of yield per area. Proper spacing promotes more vigorous plant growth and intensive production over less area [8].

There are different kinds of small size seeds among those we have chosen teff seed because of the following reasons: -

* Widely used grain in Ethiopia
* Based on the recommendation given from agricultural research area
* Currently available planters have high consumption
* Teff grain is usually sowed in muddy farmlands that brings difficulty for farmers and vehicles
* After sowing Teff seeds it is not mandatory to cover the seed.

## **Statement of the problem**

Now days, many agricultural researches are showing that sowing seed in a row enhance yield productivity and has many more advantages. But in our country the implementation has many challenges. From the data found from Debre Brihan Agricultural center and Awash agricultural farm many farmers use traditional seed sowing mechanism called broadcasting and row wise sowing mechanisms such as by using plastic jars, machineries attached with animals or human.

Currently there are two widely used ways for implementing row sowing activity. The first one is by using plastics bottle which is used as a seed container. This can be visualized as, a person with the plastic bottle which contains teff seeds and sowing is performed single row at a time. Even though sowing in row wise has many advantages, its implementation mechanism is facing the following challenges. This mechanism is very tedious and time consuming, does not give the proper row spacing and it needs more labor force and it results in a back ache.

The second mechanism is by using some mechanical ground vehicles which requires the use of human or animal force for the movement and seeding. But these ground vehicles are not adaptable with the environment. These results difficulties to work with them in environmental conditions such as muddy environments and performance depend on geographical area, and row spacing is not precise. With these limitations’ farmers are forced to use traditional seed sowing mechanism which is broadcasting.

## **Purpose of the project**

This project intends to apply the achievements made by modern technologies in the improvement of *Teff* yield and farmers’ level of perception towards the application of modern *Teff* production. It also solves the challenges that farmers faced with the application of row planting of *Teff* seed. The aim of this project is to design and develop electrically operated *Teff* sowing Agrobot which can be controlled remotely with its corresponding software. This will improve farmers’ adoption level of modern agricultural technologies and enhance *Teff* yield production. The approach of this project is to automate manual and animal operated row sowing machines by developing an automated *Teff* sowing robot. The Agrobot puts the seeds at desired row and its main objective is to increase productivity, minimize the work of farmer and also reduces the time for seed sowing operation.

## **Objective**

### General objective

The general objective of this system is to design and develop *teff* sowing agrobot that will sow *teff* in agricultural field with row manner and it can be controlled from PC station with its controller software.

### Specific objective

In order to achieving the general objective, the following specific objectives will be achieved:

* Study the existing systems
* Select energy source for the robot
* Design electrical control and communication system
* Select an appropriate and easy software tool to implement agrobot
* Design and analyze seed sowing system
* Demonstrate the working model of Agrobot
* Design mechanical structure of the robot parts
* Implement using the selected tools
* Test the system
* Deploy the system

## **Feasibility Study**

The main concept here is that determining whether a proposed system can be made or achieved i.e. checking the acceptance of the system, formally it is deciding the phase of acceptance. We analyze three different types of feasibility test.

Farmers’ adoption level of modern production system of *Teff* row planting with reduced seed rate is affected by the characteristics of household, access to extension service, level of training provided, provision of row planting mechanization and above all the application cost of row planting method on teff production. With current manual row planting system as the number of house hold increases, the application level of manual row planting also increases whereas small household size cannot provide enough amount of labor from the family their application level of row planting technology was very low. Also, some households with large family size did not apply teff row planting due to fear of crop failure from family members especially children [4]. In order to make Agrobot feasible, we have to meet with the above feasibility issues.

### Operational Feasibility

Operational feasibility test is making sure if the system works with less difficulties when developed and installed. Our system can be manipulated little knowledge and human effort. Agrobot will have very simple procedure and user friendly.it will be designed such that both software user interface and hardware controller and user interface are very easy to understand by anyone. Also, our system doesn’t require much extension service since it will not require much training and there will be material like Manual which guides the user on how to control and use the Agrobot. This project is surely operationally feasible because the controller software and the robot are a good solution maker of the existing problem.

### Technical Feasibility

Technical Feasibility study is about evaluating if the current technology has a potential to develop or unable to support our proposed system. Implementation of the system will use windows operating system, programming language for desktop application and embedded system. By Assuming required hardware and software resources are available for the development and implementation of proposed system. Therefore, it is technically feasible.

### Economical Feasibility

This is to determine the benefits and savings that are expected from a proposed system and compare them with costs. If benefits outweigh costs, then the decision is made to design and implement the system. It is must accurately weigh the cost versus benefits before taking an action.

**Financial benefits**

Manual row planting is a labor-intensive activity. Manual row planting requires extra labor that is mainly because of the seeds are carefully sown. Row planting one hectare of teff land on average takes 138 person-hours [9]. Our system will reduce this number of labors with 32-person 30 minutes. Mechanical devices attached with tractor require lot of operating and maintenance cost. Also, this method is fuel-based system which is costly compared to electricity-based system. Since our system also can use solar power as main power source, this also makes it environmentally friendly by using renewable energy as well as decreasing fuel consumption of the country+.

**Hardware development cost**

Table 1 Average cost analysis of Agrobot

|  |  |  |
| --- | --- | --- |
| Item no | Materials | Average cost |
| 1 | Microcontroller | 500 |
| 2 | Resistor | 100 |
| 3 | Capacitor | 50 |
| 4 | DC motor | 50 |
| 5 | Steeper motor | 150 |
| 6 | Optical sensor | 200 |
| 7 | Wi-Fi module | 400 |
| 8 | Metal bar | 500 |
| 9 | Mechanical part of robot | 500 |
| 10 | Other costs | 200 |
| Total | | 2650 |

## **Scope and limitation**

### **Scope**

This project will mainly focus on sowing seed with the prepared land. This robot has no digging and soil covering part. So, the farmer is expected to prepare the land for sowing. Eventually after the sowing operation is completed, covering the soil is done manually. This system is intended to allow users to seed sowing in row, view how much farm land is covered, user can check how much seed is in seed container, user can communicate with the robot with the help of Android application, and user can control the movement of the robot.

### **Limitation**

The Agrobot is limited by the following features:

* The bar on which the robot moves is controlled or moved by the person
* Life time of battery

## **Significance of the project**

* It maintains the proper row spacing.
* Seed rate can be controlled.
* Due to small size machine is portable
* Cost efficient, save energy (in perspective of human and energy used by tractors), money and time of users.
* Reduce the manual work.
* Less skilled technicians are sufficient to operate.
* It will enhance the habit of row planting.
* It will increase productivity of *Teff.*
* It will save seed by avoiding unwanted seed loss.
* It will be used as reference for other studies.
* It will generate report.
* It will enhance farmers’ level of perception towards the application of modern *Teff* production.

## **Methodology**

Data source

The first and the most valuable data source for our system is agricultural research centers which work on agricultural Engineering mechanization and zonal agricultural offices.

### Fact finding methodology

**Primary sources**

1. Interviews

At the time we have visited Awash Agricultural Institution, we were able to interview some agricultural mechanization engineer about their recent model, usability and feasibility and also some future improvement they are going to apply.

2. Observation

After we had a brief knowledge about row planting mechanism, we were able to see currently available *Teff* row planter mechanical device found in awash agricultural institution. We were able to understand the basics of seed rate controlling mechanism.

**Secondary source**

1. Document Analysis

It describes the act of reviewing the existing documentation of comparable business processes or systems in order to extract pieces of information that are relevant to the current project, and therefore should be consider projects requirements. We have referred and analyzed different documents about currently available working model and design of agricultural seed sowing robot.

### System Development Methodology

A system development methodology refers to the framework that is used to structure, plan, and control the process of developing the system. There are different types of system development methodologies to develop Seed sowing Agrobot; from those options we will use agile method. Because agile method is an iterative, team-based, communicative with the users, time-boxed approach development nature. This approach emphasizes the rapid delivery of an application in complete functional components. Rather than creating tasks and schedules, it works by packaging activities into phases called “sprints.”

Within each sprint, we use four different phases those are: -

* Requirement gathering: understanding and identifying requirement of the system (ideas from the users)
* Documentation preparation phase
* Implementation phase
* Tasting phase

### Development Tools

Table 2 Client and server-side tools

|  |  |
| --- | --- |
| Tools | Used for |
| Android | To develop the Android application |
| C language | For the embedded system parts |
| MYSQL | For backend database |
| PHP | Server-side programming |

Table 3 Other tools

|  |  |
| --- | --- |
| Tools | Application |
| MS-word | For writing the documentation |
| MS-ppt | For preparing presentation slides |
| Edraw-max and enterprise architecture | For drawing the UML diagrams |
| MicroC | For programming the embedded systems |
| Arduino IDE | For programming Arduino |

**Hardware Requirements**

* **Micro-controller**: is a small computer on a single [i](https://en.wikipedia.org/wiki/Integrated_circuit)ntegrated circuit and used in automatically controlled products and devices.
* **Resistor** and **Capacitor**: this component play an important role in how an electronic circuit behaves, with each connected by conductive wires through which electric current can flow.
* **Stepper Motor:**
* **Optical sensors or IR:** electronic detectors that convert light, or a change in light, into an electronic signal and can be used to measure distance.
* **Wi-Fi module**: for communication
* **Ultrasonic sensor (optional):** measure distances based on transmitting and receiving ultrasonic signals.
* **Horizontal metal plate:** metal plate on which the Agrobot moves.
* **Metal for making our robot**
* **Plastic tube** for stack the seed
* **Circle Wood or metal** for holding seed
* **Plastic container** for holding whole seeds
* **Personal computer:** almostall of our project phases (documentation and coding) are done by computers
* **Flash**: required for data movement
* **Stationeries (pen, paper):** for writing all necessary documentations associated with the project
* **Note book:** to take notes during data collection and for other documentations

## **Constraints and Assumptions**

### Constraints

* Ease of use: seed should be free from dirt, since our system does not have complex seed sowing activities like dirt separation mechanism form seed

### Assumptions

* There is prepared land which is going to be sowed
* user has basic knowledge about manipulation of robot

## **Testing plan**

### Unit test

At this phase of testing we will test the individual units/ components of software to validate that each unit of the software performs as designed and eliminate faults in procedure and functions point of view by using black box and white box testing.

Tasks we will perform under this phase are:

* Unit test plan
* Identify the unit test objectives
* Prepare test cases that includes information such as set of test inputs, execution condition and expected output
* Perform the tests according to our plan
* Analyze the test results
* Document the test results

### Integration test

At this phase of testing, we will combine the individual units and taste them as a group and this is performed after the unit testing.

The following tasks are going to be performed under this phase: -

* Prepare integration test plans
* Identify integration test objectives
* Identify integration test acceptance criteria
* Perform the tests according to our plan
* Document the test results

## **Overview of project phase**

The project phases are:

* Requirement gathering and Analysis: This phase begins with analyzing what exactly the system has to do. The system overview helps see the big picture of the project and understand which steps need to be carried out. The product of this stage is the general system requirements. This document will be modified as the project is undertaken.
* Design: in this phase the overall design of the system will be performed.
* Implementation: The goal of this phase is building the target system based on the specifications developed in the previous phases. Transferring the specification algorithms into a programming language
* Testing: is an investigation conducted to provide stakeholders with information about the quality of the product or service under test.
* Maintenance: The task of this phase is the proper functioning of the system. To improve a product or system, it should be continuously maintained. Software/hardware maintenance involves detecting and correcting errors, as well as extending and improving the software/hardware itself.

## **Schedule**

Table Schedule

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Phases | Month | | | | |
| Nov 12-2018-  Feb 2-2019 | Feb 13 2019-  Mar 25-2019 | May 10-2019-  Jun 30-2019 | July 01-2019-  July 10-2019 | Jul-15-2019-  Jul-22-2019 |
| Requirement gathering and Analysis |  |  |  |  |  |
| Design |  |  |  |  |  |
| Implementation |  |  |  |  |  |
| Testing |  |  |  |  |  |
| Maintenance |  |  |  |  |  |

## **Team Composition**

Table 5 Team composition

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No | Name | ID | Email | Responsibility |
| 1 | Mengistu Abebe | R/00688/07 | [ethiomengistu@gmail.com](mailto:ethiomengistu@gmail.com) | Coordinator, Design, Analysis, programming and  Implementation |
| 2 | Merga W/yohannes | R/00691/07 | mergawolde[@gmail.com](mailto:adnanyodarge@gmail.com) | Design, Analysis, Programming, Implementation and Testing |
| 3 | Getu Lakew | R/00466/07 | Getulakew26[@gmail.com](mailto:abeleshetu@gmail.com) | Requirement Gathering, analysis, design, implementation and Testing |
| 4 | Tizita Daba | R/00958/07 |  | Requirement Gathering, analysis, design, implementation and Testing |
| 5 | Abdin Tekalegn | R/00009/07 | [Abdisatekalign23@gmail.com](mailto:Abdisatekalign23@gmail.com) | Requirement Gathering, analysis, design, implementation and Testing |

# CHAPTER TWO

# CURRENT SYSTEM

## **Major Functionality of the Current System**

The major functionality of the current system is to sow *Teff in row manner*. It can be a plastic jar hold with human or mechanical machineries attached to vehicles or animal.

* Row planting with plastic bottle will follow the following process
* Land will be prepared
* The line to keep the row straight will be prepared using rope.
* Usually half litter plastic bottle is prepared.
* The bottle cover is pined with small nail
* Fill about the half of the bottle with Teff seed
* Upside down the bottle and shake well to drop the seed with following the line
* One person can sow seed one row at a time
* Using mechanical attached device with animals
* Land will be prepared
* Seed container will be filled with Teff seed.
* Device will be strongly attached with body of animal or vehicles.
* The farmer will guide the animal (Ox or donkey) or drive the vehicle to desired direction.
* Mechanical device will be pulled by animal or vehicle and seed is dispensed.
* Can sow 4 rows at a time

## **User of the Current System**

Currently the Users of the system are:

* Farmer
  + Small land holder exercises these manual row planting
* Research centers
  + Currently Agricultural institutes are using the above methods of row planting to assess the effecting of row planting

## **Problem of the Current System**

We are in the era of automation where people getting used to adopt Smart techniques to get their work done. One of the largest and interesting automation sectors is agricultural sector used for sowing seed in row. Despite agriculture being one of the most Important fields for determining the growth of a country, it is lagging in terms of smart Sowing techniques and equipment’s have kept on progressing. But in our context most Ethiopian farmers are facing difficulties with manual row planting of *Teff.* This mechanism has many limitations compared to our system.

In manual row planting using plastic jar, seed rate controlling mechanism is guided with hand pressure applied to the bottle. The person, sowing the seed, should always be tilted to ground and squeezing the plastic jar all the way till sowing is finished. For be best result the person should strictly follow the row line and expected to apply constant squeezing pressure. The difficulties here are that it is very difficult to keep track of the row line and maintain constant squeezing pressure for longer distance. Since Teff seeds are very small size it is very hard to dispense seed with bottle cap hole. With keeping such condition one person requires a lot of time to sow small land. Not only this but also tilting near to ground and walking longer distance caused a series back ache. This will reduce average working time of one person, eventually to implement this manual row planting with plastic mechanism one needs larger number of labor force. One-person sow one row at a time

There are also some mechanical 2-wheel vehicle machineries attached with animals or vehicles. Even though this mechanism decreases some labor force, pulling this vehicle with animal or vehicle requires a lot of effort since the environment is muddy most of the time. Due to this, such vehicle experiences problem with movement on the mud and most of the time they are stuck in the middle of mud. With such difficulties manipulating direction of vehicles and controlling row spacing is very difficult. This approach requires also a large human effort in addition to animals (ox or donkey) or vehicle due to difficulties for movement and direction manipulation.

## **Business role of the systems**

In order to use *Teff* sowing Agrobot (TSA) we have stated some business rule

* User should send request to nearby kebele for using this service.
* User should have basic knowledge of manipulation robot.
* Agrobot software has full access and control of the robot.
* Agrobot has embedded with some controls.
* User is expected to have well prepared land which is ready to be sowed.
* User is expected to buy or rent *Teff* sowing Agrobot.
* User is expected to buy license for using Agrobot software.

# CHAPTER THREE

# PROPOSED SYSTEM

## **Overview**

Agricultural robots automate slow, repetitive and dull tasks for farmers, allowing them to focus more on improving overall production yields. In our country agricultural mechanization, that is to say the replacement of human or animal effort with machine power has now proceeded to the point where most tasks in agriculture in the development world are mechanized. In most, control of the machine is carried out human operator rather than automatically [2].

The Agrobot that we are going to develop will help Ethiopian farmers to exercise *Teff* row planting which enhance productivity of farmer with the same given input and farm land. The robot is small and electrically operated so that there is no need of mechanical support from human or animal. It provides row spacing, speed controlling, and coverage calculation mechanism. Also, time required for sowing operation minimized. This will decrease the number of labor force needed compared to currently available row planting mechanism. Agrobot can record the necessary data and we can use this data with our software to generate report.

## **Functional Requirement**

*Teff* sowing Agrobot will perform the following functionalities:

* Communicate with Agrobot software
* adjust the row spacing
* Determine the level of seeds in the seed container and send to Agrobot software.
* Notify to the user when critically low seed is available
* move back and forth on horizontal bar (x or y axis)
* Sending data to Agrobot software how much farm land was covered by seed
* Take input how much distance to sow seed on the horizontal bar

Agrobot software also has the following functional requirement

* User can login and logout to the app
* User can create account
* Establish communication with Agrobot
* control the speed of robot
* view the level of the seed in the seed container
* view the position of robot
* view how much farmland covered by the seeds
* Give direction of robot movement
* Give input distance to sow.
* generate a report
* display notification when critical low seed level is reached

## **Non–Functional Requirement**

These are the constraint on the services or functions offered by the system. These also in compose time constraint on development process and standards. They are more critical than individual requirement. So, we specified the following major emergent properties of Agrobot: -

**Usability**

The system should be easy to learn and understandable for the user. Since system will be developed by considering all farmers and agriculture research centers. It is easy to use and learn. In addition, it will have a user manual that tells the user how to use the system and we will give enough training for the users on how to operate the system and even it will be easier for users to understand hence we will be using native languages on the manual and the application developed.

**Reliability**

The system should be highly reliable. Since the user of the system is agricultural research centers and farmers, the system will be best in handling errors and display an appropriate error message in order to minimize system failures.

**Maintainability**

The system should be maintainable. Because the interaction between subsystems will be loosely couple and the interaction between classes and operations will be highly cohered, changes made on our system such as adding other functionality shouldn’t affect the existing functionality of the system.

**User interface**

From each and every page the user can easily go to any desired page. There will be an absolute and relative linking.

* The design or layout of every form will be very clear to the user.
* In the login page the user can easily enter the desired password and username. Then it will give the successful log in message or error message and allow them to try again if either user name or password or both of them are not correct.

**Performance consideration**

The performance of the System can be seen as aspects of:

* Response time: The system should give the right response for the User as fast as possible, since the system we are developing is Android application. So, it will not be sophisticated to use and it will not take much time to handle responses.
* Workload: The workload of the robot can calculate relative to the speed of robot.
* Accuracy: Agrobot is accurate when sowing the seed and spacing between consecutive rows.
* Scalability: It can work on small- and large-scale farms
* Availability: The seed sowing agriculture robot is reliable enough to sustain in any condition. It gives consistently correct results, so User can get information from the robot continuously while the robot is doing its job.
* Efficiency: Seed sowing Agrobot will utilize processor capacity, disk space and memory efficiently.

**Error Handling and Extreme Conditions**

* Each error that may occur in Agrobot will be handled accordingly in order to reduce the amount of failure. Since users of our system are human, they may make mistakes, each and every input box is going to be handled according to their type.

## **System Model**

### **Scenario**

**Scenario 1:** for login

**Initial assumption**: users must have the application and install in the appropriate device.

**Normal**: The system displays the login page to user. Then the user enters the username and password to login form. After filling the form user clicks the login button and if the user enters the correct password and username the user will be redirected to home page.

**What can go wrong**: If user enters wrong username and password system prompts error message to enter the password and the username again on the login page.

**System state on completion:** System will save the information to the database

**Scenario 2:**  for Agrobot determine critical seed level

**Initial assumption:** sensors must be properly mounted and configured so that it can sense properly and communication must be established between the Agrobot and the Server.

**Normal:** The seed level sensor continually senses the seed level in the container and sends its data to the microcontroller, so that the seed level is determined and the Agrobot will automatically notify when critically low-level seed is available in the container to the User.

**What can go wrong:**

**System state on completion:** The Agrobot will notify to the user when the seed level reaches its critical level by saving the necessary data to the database then to the application.

**Scenario 3:** for user maintain row spacing

**Initial assumption:** user knows the standard row spacing distances.

**Normal:** User when starting to sow at first goes to the Agrobot and manually adjusts the spacing between the rows with the desired standard spacing distances.

**What can go wrong:**

**System state on completion:** the agrobot will maintain the spacing between rows.

**Scenario 4:** for agrobot determine farmland coverage

**Initial assumption:** communication must be established between the Agrobot and Server. And also, Sensors must be properly mounted and configured so that it can sense properly.

**Normal:** Distance sensor continuously send data of the coverage’s to the microcontroller and the microcontroller calculates the distance covered so far and sends the data to Server.

**What can go wrong:**

**System state on completion:** the necessary data will be saved to the database.

**Scenario 5:** for agrobot determine the level of seeds

**Initial assumption:** Connection must be established between the Agrobot and Server. Also, Level sensor must be properly mounted and configured.

**Normal:** Level sensor senses the level of seeds in the container and gives the data to micro controller. After that the microcontroller sends the levels of the seeds to the Server.

**What can go wrong:**

**System state on completion:** Determines the levels of the seeds available and the necessary data will save to database.

**Scenario 6:** for agrobot and user determine the input distance

**Initial assumption:** User must have the Agrobot software and authentication. Also, Communication must be established between the Agrobot and Agrobot software and Sensors must be properly mounted and configured so that it can sense properly.

**Normal**: The user must login to the system and chooses determine input distance to sow from the menu and enters the input distance to sow in to the forms. After that the Agrobot receives the entered distance from database and then the agrobot will sow depending on the input distance.

**What can go wrong:** If input distance is not in a valid range then the system will prompt error message to user.

**System state on completion:** the agrobot will maintain the input distance to sow and the necessary data will save to database.

**Scenario 7:** for user control speed

**Initial assumption**: User must have the Agrobot software and authentication. Also, communication must be established between the Agrobot and Agrobot software.

**Normal**: The user must login to the system and chooses speed control option from the menu and enters the speed rate using seek bar. After that the Agrobot receives entered speed rate from database then agrobot will control the speed of robot to the specified rate.

**What can go wrong:**

**System state on completion:** the agrobot will achieve desired speed.

**Scenario 8:** for user control movement

**Initial assumption**: User must have the Agrobot software and authentication. Also, communication must be established between the Agrobot and Agrobot software.

**Normal:** The user must login to the system and choose movement control option from the menu and select movement direction that is either back or forth. After that the Agrobot receives entered movement direction from database then agrobot will control the movement direction of robot as specified.

**What can go wrong:**

**System state on completion:** The new direction of Agrobot is set.

**Scenario 9:** for agrobot control movement

**Initial assumption**: Controlling components must be configured correctly.

**Normal**: Connection between the sensor and controller must be established and when the user presses input key to the desired direction it will be sent to the microcontroller and based on the input the Agrobot will set to the new direction.

**What can go wrong:**

**System state on completion:** The new direction of Agrobot is set.

**Scenario 10:** User for Generate report

**Initial assumption**: User must have the Agrobot software and authentication. Also, Communication must be established between the Agrobot software and Server and Agrobot history should be logged in database.

**Normal:** The user must login to the system choose generate report option from the menu and select time or days to generate report. After that the Agrobot software will fetch data from the database to generate the report.

**What can go wrong:** If there is no data to be fetched then the system will display error message to user.

**System state on completion:** The agrobot software will generate the required information’s.

**Scenario 11:** for userView position

**Initial assumption**: User must have the Agrobot software and authentication. Also, Communication must be established between the Agrobot software and Server.

**Normal**: The user must login to the system and choose position of robot option from the menu and the user clicks robot position. After that Agrobot software fetches position data from the database and displays the position of robot.

**What can go wrong:**

**System state on completion:** Thecurrent Position of Agrobot can be traced.

**Scenario 12:** user view farmland covered

**Initial assumption**: User must have the Agrobot software and authentication. Also, Communication must be established between the Agrobot software and Server.

**Normal**: The user must login to the system and choose view option from the menu and after that the Agrobot software fetches data from the database and displays farmland covered by the seed.

**What can go wrong:**

**System state on completion:** Farm land covered will be displayed.

**Scenario 13:** for userView seed level

**Initial assumption**: User must have the Agrobot software and authentication. Also, Communication must be established between the Agrobot software and Server.

**Normal**: User must login to the system and choose view option from the menu and After that Agrobot software fetches data from the database and displays the level of seed.

**What can go wrong:**

**System state on completion:** The level of seeds will be displayed.

**Scenario 14**: for userview critical seed level

**Initial assumption**: User must have the Agrobot software and authentication. Also, Communication must be established between the Agrobot software and Server.

**Normal:** The user must login to the system and the Agrobot software fetches the data from the database and notifies when critical level of seed in the container is reached.

**What can go wrong:**

**System state on completion:** warning notification about seed level displayed

**Scenario 15:** for user Logout

**Initial assumption**: User must have the Agrobot software.

**Normal:** from the current page user can clicks logout button then the user leaves the page and will be redirected to the login page.

**What can go wrong:**

**System state on completion:** The user must be leave from the page.

**Scenario 16:** for userforRegistration

**Initial assumption**: User must have the Agrobot software

**Normal**: user must have the application and user should install the application. System displays default page then user select the registration option and User fills the registration form. If the form is correct system will display successful message.

**What can go wrong:** If user enters wrong data system prompts error message and redirect to registration page.

**System state on completion:** when user fills the forms correctly then the necessary data will save to database.

**Scenario 17:** for user and agrobot maintain input distance

**Initial assumption**: communication must be established between the Agrobot and Agrobot software. Also, Sensors must be properly mounted and configured so that it can sense properly.

**Normal**: User initiates the agrobot and enters the row spacing distance via keypad then microcontroller receives the entered and Agrobot maintains input distance.

**What can go wrong:** If input distance is not valid then the system will prompt error message.

**System state on completion:** Required amount of farm land will be covered with seed.

**Scenario 17:** for sowing

**Initial assumption**: Sensors must be properly mounted and configured so that it can sense properly.

**Normal**: the user initiates the agrobot and order to sow on the prepared field.

**What can go wrong:**

**System state on completion:** sows the seeds on the desired field.

## **Use case Model**

### Actor Identification

The actors of our system are:

* User
* Agrobot

### Use case diagram and Description

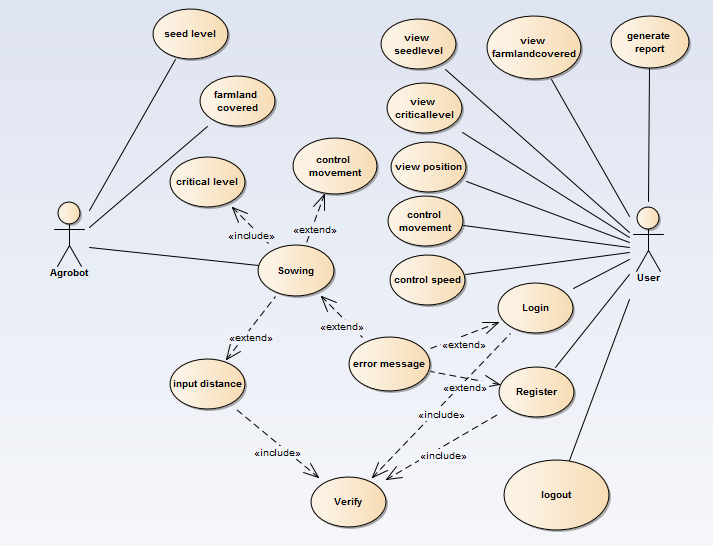


Figure 1 Use Case diagram

**Use case Description**

Table 6 use case description of login

|  |
| --- |
| Description 1 |
| Use case name | Login |
| Use case number | Uc01 |
| Description | Authentication to the system |
| Participating actor | User |
| Pre-condition | * + Must have the application and account. |
| Flow of event | * + - 1. The user must have the application.       2. The user opens the application.       3. System displays homepage with login.       4. User enters the username and password to login.       5. If the password and username are correct system displays homepage. |
| Post condition | System will save the information to the database. |
| Alternative flow of event | If user enters wrong username and password system prompts error message and redirect to login page (redirect to step 3). |

Table 7 use case description of Notify critical Seed

|  |
| --- |
| Description 2 |
| Use case name | critical Seed level |
| Use case number | Uc02 |
| Description | Notifies when seed in the container reaches its critical level |
| Participating actor | Agrobot |
| Pre-condition | * Sensors must be properly mounted and configured so that it can sense properly. * Communication must be established between the Agrobot and Server. |
| Flow of event | 1. Seed level sensor sends data to microcontroller. 2. Seed level is determined.   3. The Agrobot will notify when critically low-level seed is available to User. |
| Post condition | Notify the message and save to database |
| Alternative flow of event |  |

Table 8 use case description of row spacing

|  |
| --- |
| Description 3 |
| Use case name | row spacing |
| Use case number | UC03 |
| Description | User maintains the row space for sowing. |
| Participating actor | User |
| Pre-condition | * User must know the standard row spacing distances. |
| Flow of event | 1. User goes to the agrobot 2. User adjusts the spacing to the standard row spacing. 3. Agrobot maintains the spacing for sowing. |
| Post condition | Maintains the spacing between rows. |
| Alternative flow of event |  |

Table 9 use case description of farmland covered

|  |
| --- |
| Description 4 |
| Use case name | Farmland covered |
| Use case number | UC04 |
| Description | Determine the farmland covered by the seeds. |
| Participating actor | Agrobot |
| Pre-condition | * Communication must be established between the Agrobot and Server. * sensors must be properly mounted and configured so that it can sense properly |
| Flow of event | 1. Distance sensor continuously sends data to microcontroller. 2. Microcontroller calculates the distance and sends the data to the Server. |
| Post condition | Data will be saved to the database. |
| Alternative flow of event |  |

Table 10 use case description of level of seed

|  |
| --- |
| Description 5 |
| Use case name | Level of seed |
| Use case number | UC05 |
| Description | Determine the level of the seeds. |
| Participating actor | Agrobot |
| Pre-condition | * Connection must be established between the Agrobot and Server. * Level sensor must be properly mounted and configured. |
| Flow of event | 1. IR sensor senses level of seed in container. 2. Sensor data given to micro controller. 3. Send the levels of the seeds to the Server. |
| Post condition | Determines the levels of the seeds available and save to database |
| Alternative flow of event |  |

Table 11 use case description of take input distance

|  |
| --- |
| Description 6 |
| Use case name | Input distance |
| Use case number | UC06 |
| Description | Agrobot will sow on the inputted distance. |
| Participating actor | Agrobot, user |
| Pre-condition | * User must have the Agrobot software and authentication. * Communication must be established between the Agrobot and Agrobot software. * Sensors must be properly mounted and configured so that it can sense properly. |
| Flow of event | 1. The user must login.  2. User clicks on input distance button.  3. User enters desired distance to sow the seeds. |
| Post condition | Required amount of farm land will be covered with seed. |
| Alternative flow of event | If input distance is not in valid range prompts error message. |

Table 12 use case description of control speed

|  |
| --- |
| Description 7 |
| Use case name | Control speed |
| Use case number | UC07 |
| Description | User can control the speed at which the Agrobot sows |
| Participating actor | User |
| Pre-condition | * User must have the Agrobot software and authentication. * communication must be established between the Agrobot and Agrobot software |
| Flow of event | 1. The user must login 2. Select desired speed of robot 3. Click adjust speed 4. Speed of robot changed |
| Post condition | desired speed is achieved |
| Alternative flow of event |  |

Table 13 use case description of control movement

|  |
| --- |
| Description 8 |
| Use case name | Control movement |
| Use case number | UC8 |
| Description | control the movement. |
| Participating actor | User |
| Pre-condition | * User must have the Agrobot software and authentication. * communication must be established between the Agrobot and Agrobot software |
| Flow of event | * The user must login * User select movement direction either back or forth * User click adjust movement * Agrobot adjust itself to desired direction. |
| Post condition | New direction of Agrobot is set |
| Alternative flow of event |  |

Table 14 use case description of control movement

|  |
| --- |
| Description 9 |
| Use case name | Control movement |
| Use case number | UC9 |
| Description | Agrobot can control the movement of the Agrobot. |
| Participating actor | Agrobot |
| Pre-condition | * Controlling components must be configured correctly. |
| Flow of event | 1. Connection between the sensor and controller must be established. 2. The user presses input key to desired direction. 3. The Agrobot set to new direction. |
| Post condition | Desired direction of movement is achieved |
| Alternative flow of event |  |

Table 15 use case description of generate report

|  |
| --- |
| Description 10 |
| Use case name | Generate report |
| Use case number | UC10 |
| Description | User can generate report on what is sowed. |
| Participating actor | User |
| Pre-condition | * User must have the Agrobot software and authentication. * Communication must be established between the Agrobot software and Server. * Agrobot history should be logged in database. |
| Flow of event | 1. The user must login.  2. The user clicks generate.  3. Agrobot software fetches data from the database. |
| Post condition | Report will be generated. |
| Alternate condition | If there is no data to be fetched it will display error message. |

Table 16 Table use case description of view position

|  |
| --- |
| Description 11 |
| Use case name | View position |
| Use case number | UC11 |
| Description | User can view where the Agrobot is. |
| Participating actor | User |
| Pre-condition | * User must have the Agrobot software and authentication. * Communication must be established between the Agrobot software and Server. |
| Flow of event | 1. The user must login.  2. Use clicks view  3. Agrobot software fetch position data from the database  4. User views the position of robot. |
| Post condition | Current Position of Agrobot can be traced. |
| Alternate condition |  |

Table 17 Table use case description of view farmland covered

|  |
| --- |
| Description 12 |
| Use case name | View farmland covered |
| Use case number | UC12 |
| Description | User can view the land covered by the seeds. |
| Participating actor | User |
| Pre-condition | * + User must have the Agrobot software and authentication.   + Communication must be established between the Agrobot software and Server. |
| Flow of event | **Flow of event**  1. The user must login.  2. User click view farmland covered.  3. Agrobot software fetches data from the database. |
| Post condition | Farm land covered will be displayed. |
| Alternate condition |  |

Table 18 use case description of view level of seed

|  |
| --- |
| Description 13 |
| Use case name | View level of seeds |
| Use case number | UC13 |
| Description | User can view the level of seeds in the container. |
| Participating actor | User |
| Pre-condition | * + User must have the Agrobot software and authentication.   + Communication must be established between the Agrobot software and Server. |
| Flow of event | 1. The user must login.  2. User clicks view.  3. Agrobot software fetches data from the database.  4. The user can view level of seed. |
| Post condition | The level of seeds will be displayed |
| Alternate condition |  |

Table 19 use case description of view critical seed level

|  |
| --- |
| Description 14 |
| Use case name | View critical seed level |
| Use case number | UC14 |
| Description | User can view the notified critical level of seeds availability in the container. |
| Participating actor | User |
| Pre-condition | * User must have the Agrobot software and authentication. * Communication must be established between the Agrobot software and Server. |
| Flow of event | 1. The user must login  2. Agrobot software fetch data from the database  3. User view notification when critical low seed level is reached. |
| Post condition | warning notification about seed level displayed |
| Alternate condition |  |

Table 20 use case description of logout

|  |
| --- |
| Description 15 |
| Use case name | Logout |
| Use case number | UC15 |
| Description | Used to leave from the page. |
| Participating actor | User |
| Pre-condition | * User must have username and password. |
| Flow of event | 1. User clicks logout link. 2. The user leaves the page. |
| Post condition | The user must be leave from the page. |
| Alternate condition |  |

Table 21 use case description of registration

|  |  |
| --- | --- |
| Description 16 |  |
| Use case name | Registration |
| Use case number | UC16 |
| Description | Adds the user to the system |
| Participating actor | User |
| Pre-condition | * Must have the application. |
| Flow of event | 1. The user must have the application.  2. The user opens the application.  3. System displays homepage with registration.  4. User fills the registration form.  5. If the form is correct system displays homepage. |
| Post condition | System will save the user to the database. |
| Alternative flow of event | If user enters wrong data system prompts error message and redirect to registration page (redirect to step 3). |

Table 22 use case description of Agrobot take input distance

|  |
| --- |
| Description 17 |
| Use case name | Input distance |
| Use case number | UC18 |
| Description | Agrobot will sow on the inputted distance. |
| Participating actor | Agrobot, user |
| Pre-condition | * + Communication must be established between the Agrobot and Agrobot software.   + Sensors must be properly mounted and configured so that it can sense properly. |
| Flow of event | 1. User initiates the Agrobot. 2. User enter input distance to sow. 3. Microcontroller receives entered distance. 4. Agrobot Adjusts its distance to sow. |
| Post condition | Required amount of farm land will be covered with seed. |
| Alternative flow of event | If input distance is not in valid go to step 2. |

Table 23 Use case description for sowing

|  |
| --- |
| Description 18 |
| Use case name | Sowing |
| Use case number | UC18 |
| Description | When initiated agrobot will sow on the prepared land. |
| Participating actor | Agrobot, user |
| Pre-condition | * + Communication must be established between the Agrobot and Agrobot software.   + Sensors must be properly mounted and configured so that it can sense properly. |
| Flow of event | * + - 1. User initiates the Agrobot.       2. User selects start (sow) button.       3. Agrobot starts sowing on theprepared land**.** |
| Post condition | The farmland will be sowed. |
| Alternative flow of event |  |

## **Object Model**

Object model is a description of an object-oriented architecture, including the details of the object structure, interfaces between objects and other object-oriented features and functions.

Table 24 Object model

|  |  |  |  |
| --- | --- | --- | --- |
| Class | Attributes | operations | Description |
| Reset password | ID  Email  Oldpassword  Newpassword | sendPasswordResetLink()  EmailValidation() | Allows the user to reset their password. |
| Users | Id  Username  Password  First name  Last name  Email  phoneNumber | Viewseedlevel()  Controlmovement()  Controlspeed()  Viewminseedlevel()  Viewposition()  InputDistance()  rowSpacing()  Register()  Login()  Logout() | Allows the user to control movement and speed of robot, create account and view seed level, min critical seed level, position. |
| Agrobot | Agrobot\_id | Levelofseed()  CriticalSeedLevel()  rowspacinig()  Controlmovement()  inputdistance() | Allow the Agrobot to determine level of seed, min seed availability, and control movement, take input distance to sow and maintain the row spacing. |
| Sensor value | Sensor\_ID  seedLevel  InputDistance  CriticalLeevel  RowSpacing  MovmentValue  SpeedValue | Send() | Allows the Agrobot to send the level of seed, critical seed level and maintained row spacing, input distance to sow, speed value and movement value. |
| FarmLand covered | FarmId  FarmDate  Area | Landcovered() | Allows the Agrobot to determine the area covered by the seeds. |
| Report | ReportId  Reportdate  Description | Generatereport() | Allows the user to generate report based on the data’s from the database sent from the Agrobot. |

## **Class Diagram**

The Class diagram captures the logical structure of the system; the classes and things that make up the model. It is a static model, describing what exists and what attributes and behavior it has, rather than how something is done. Class diagrams are most useful to illustrate relationships between classes and interfaces.

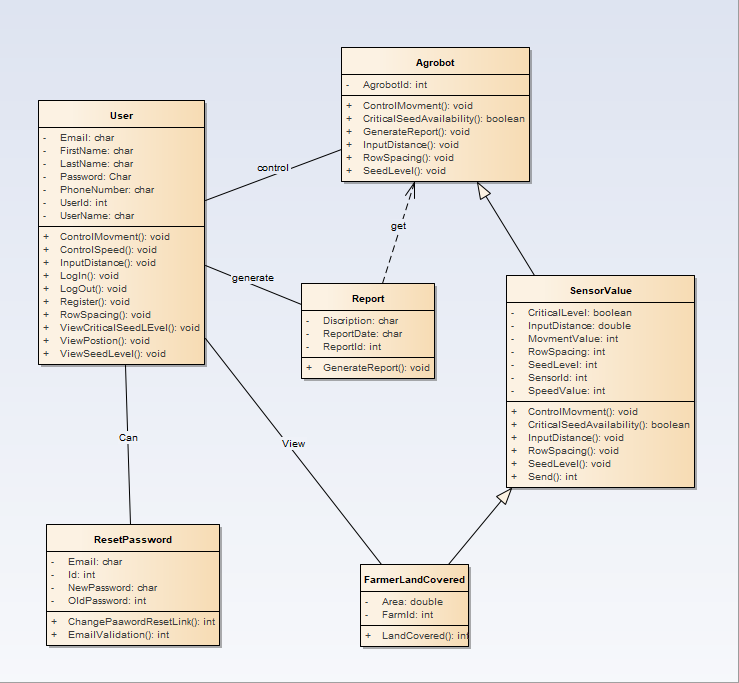


Figure 2 Class diagram for Teff seed row sowing Agrobot

## **Dynamic model**

### Sequence Diagram

A sequence diagram in a UML is a kind of interaction diagram that shows how processes operate with one another and in what order. A sequence diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. It shows, as parallel vertical lines (lifelines), different processes or objects that live simultaneously, and, as horizontal arrows, the messages exchanged between them, in the order in which they occur. This allows the specification of simple runtime scenarios in a graphical manner.

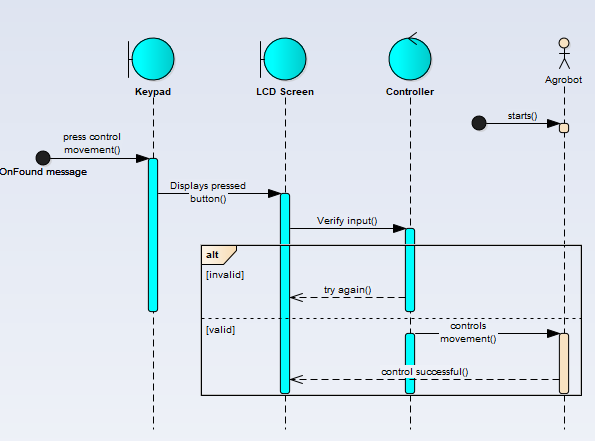


Figure 3 sequence diagram Agrobot control movement

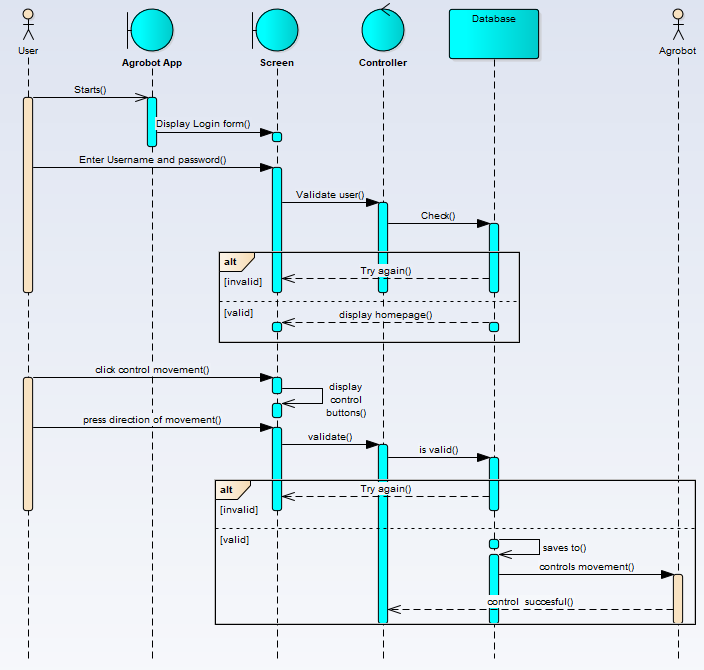
**

Figure 4 sequence diagram User control movement

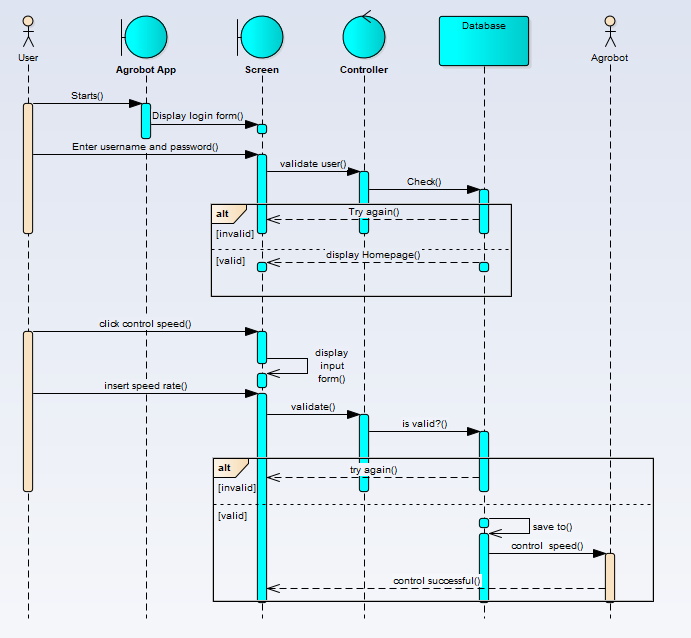


Figure 5 sequence diagram User control speed

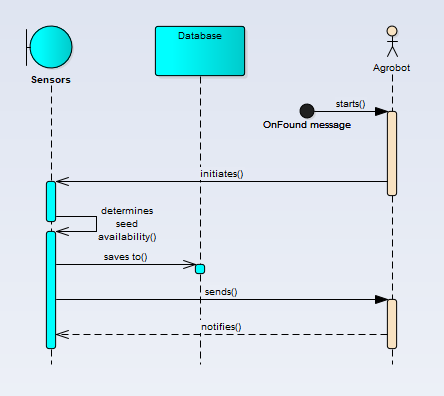


Figure 6 sequence diagram Agrobot determine critical seed level

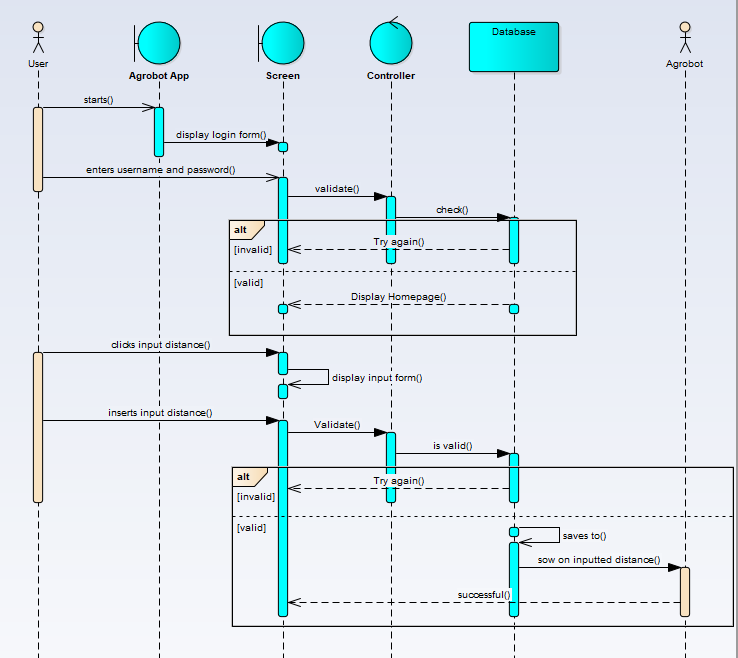


Figure 7 sequence diagram user enters input distance to sow



Figure 8 sequence diagram Agrobot determine seed level

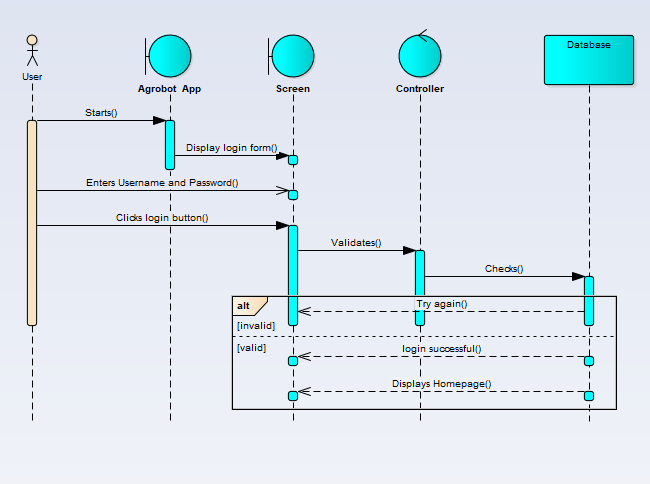
**

Figure 9 sequence diagram user login to Agrobot

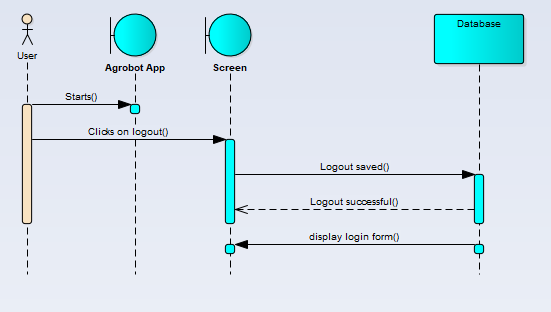


Figure 10 sequence diagram user logout

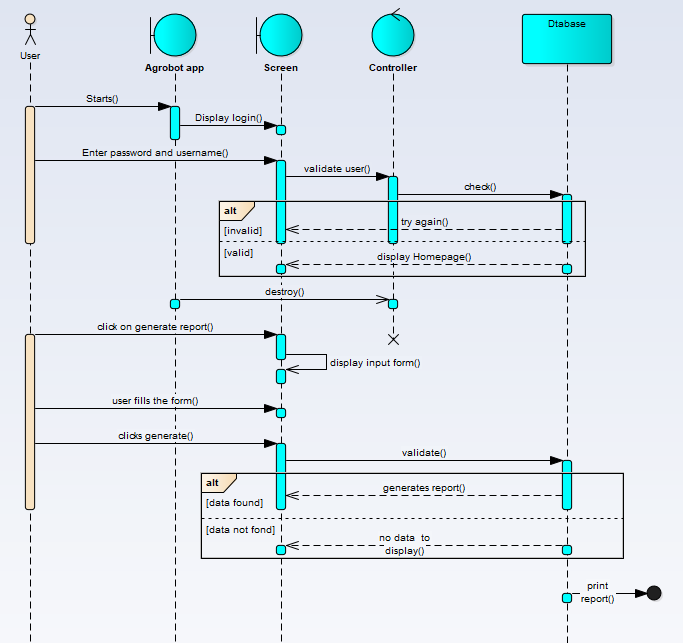


Figure 11 sequence diagram user generate report

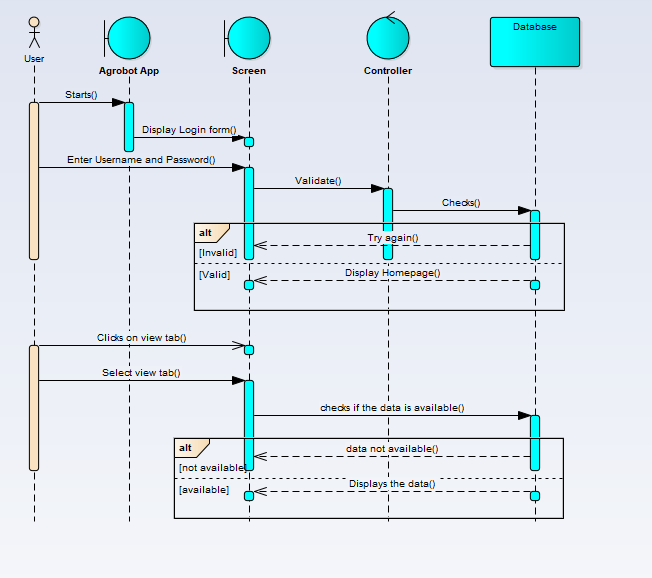


Figure 12 sequence diagram user can view details

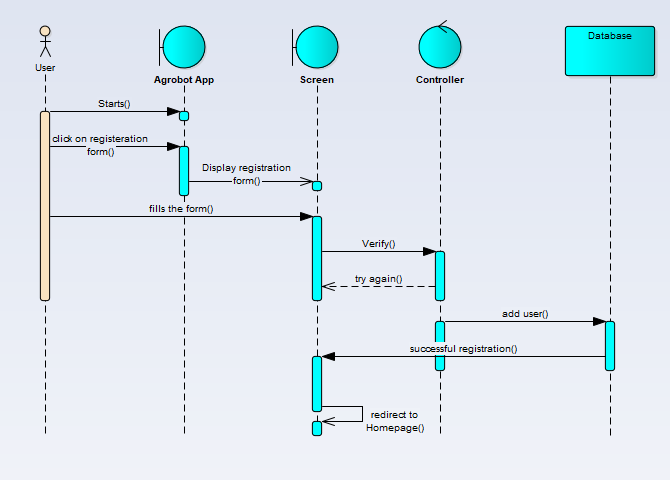
**

Figure 13 sequence diagram for registration

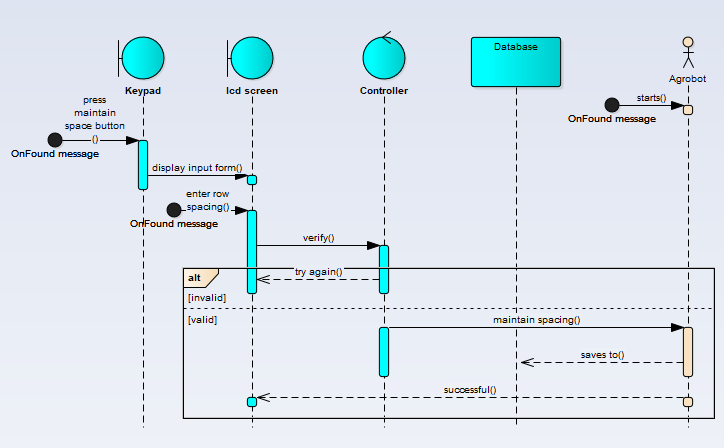
**

Figure 14 sequence diagram for Agrobot control input distance

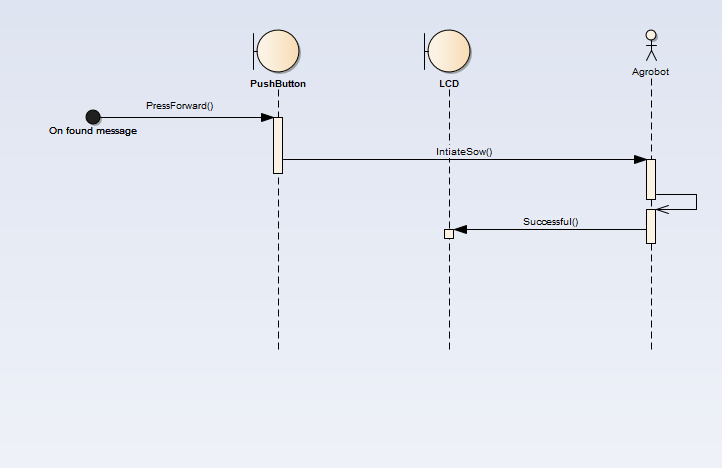


Figure 15 Sequence diagram for agrobot can sow

## **Activity Diagram**

An activity diagram describes a system in terms of activities. Activities are states that represent the execution of a set of operations. The completion of these operations triggers a transition to another activity. Activity diagrams are similar to flowchart diagrams in that they can be used to represent control flow (i.e., the order in which operations occur) and data flow (i.e., the objects that are exchanged among operations).

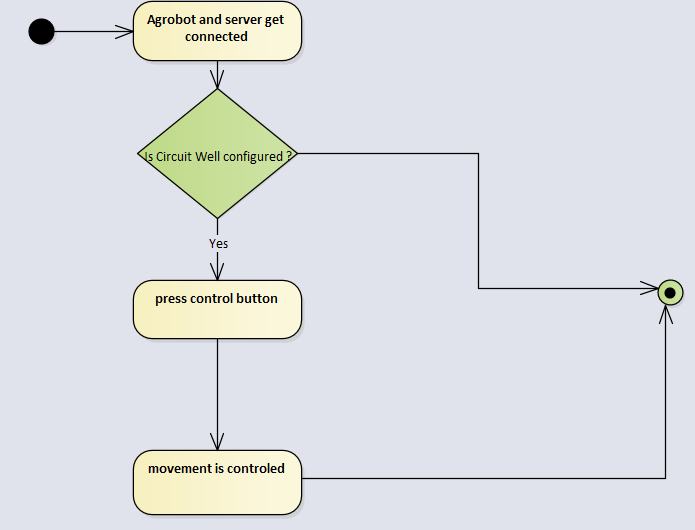


Figure 16 activity diagram Agrobot control movement

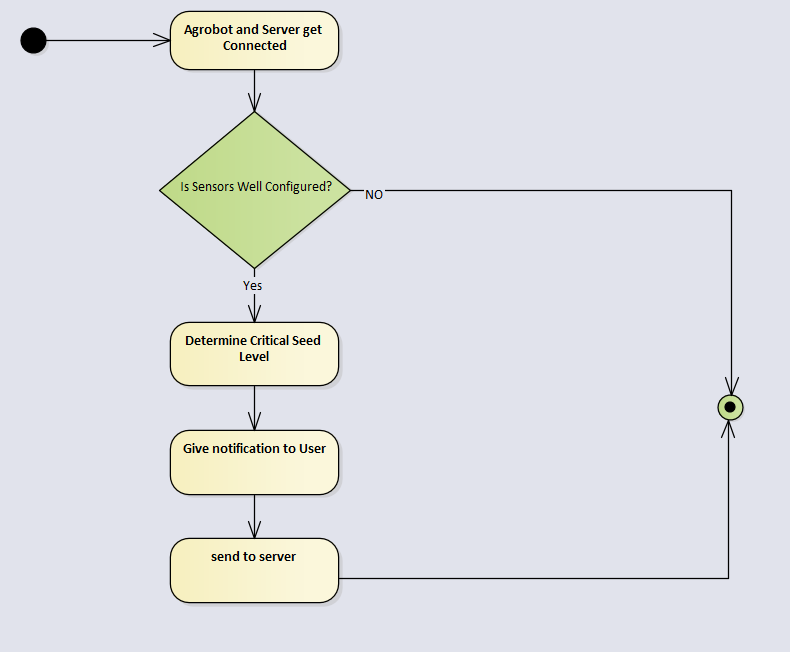


Figure 17 activity diagram Agrobot determine critical seed level

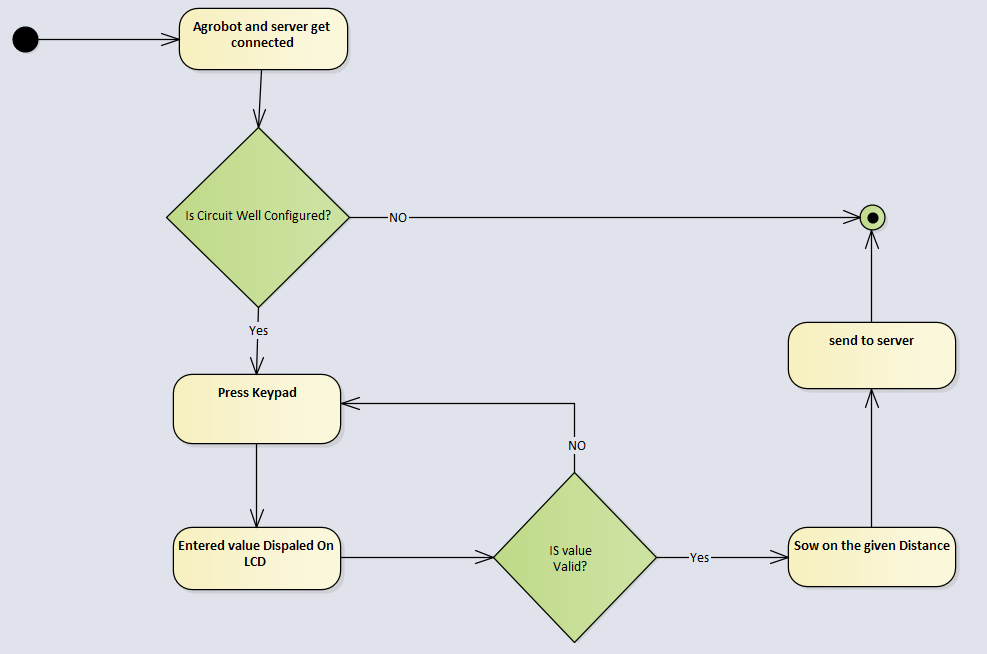
**

Figure 18 activity diagram Agrobot determine input distance to sow

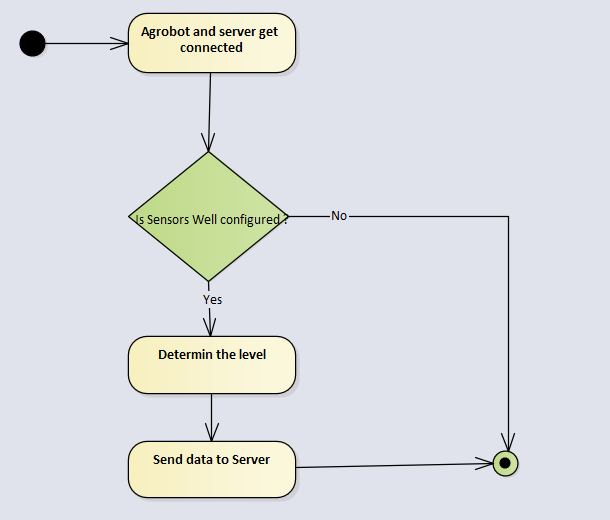


Figure 19 activity diagram Agrobot determine level of seed

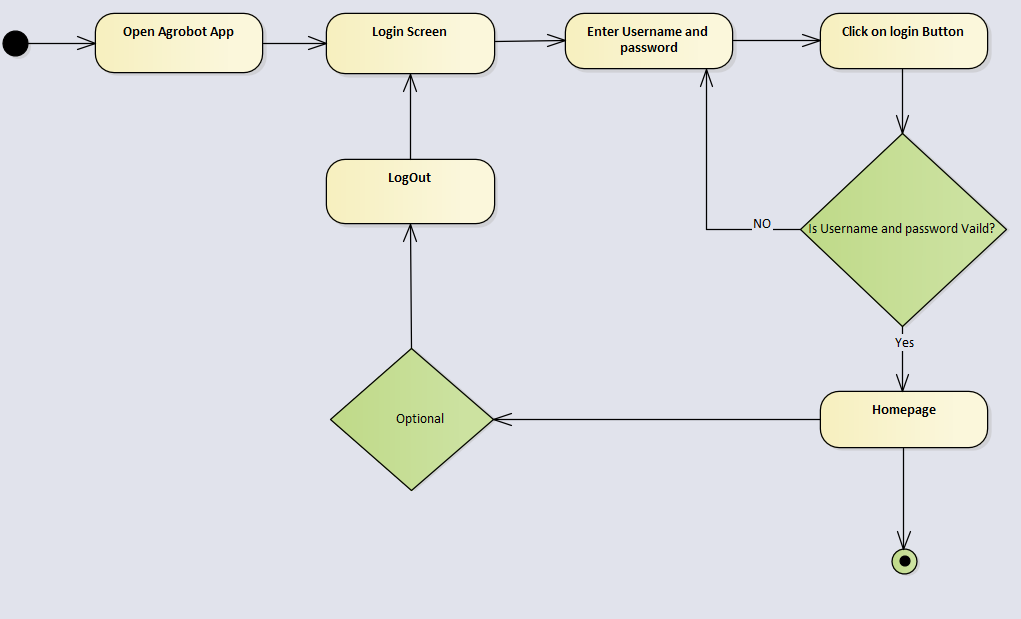


Figure 20 activity diagram user login to Agrobot

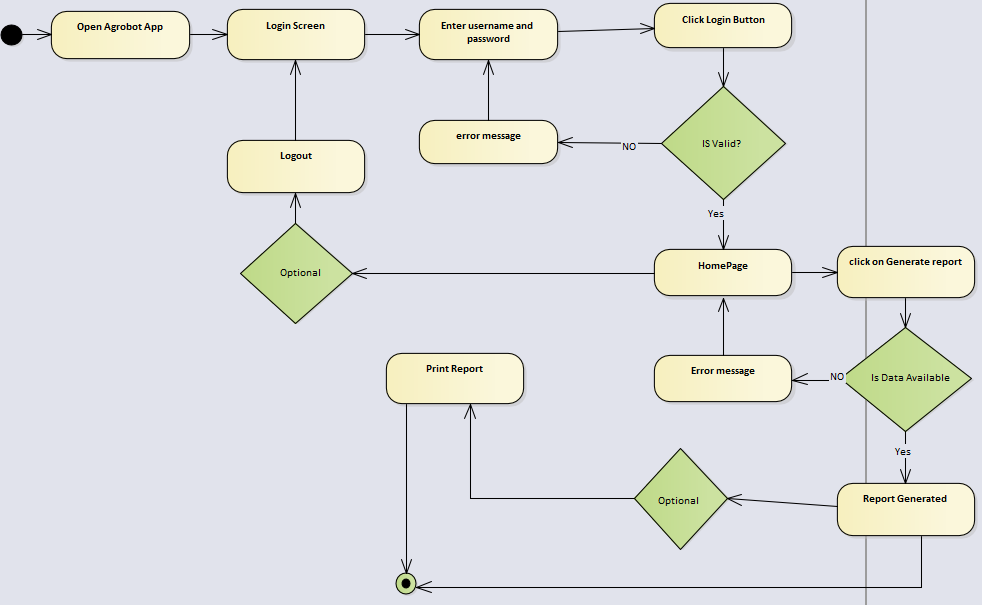


Figure 21 activity diagram user generate report

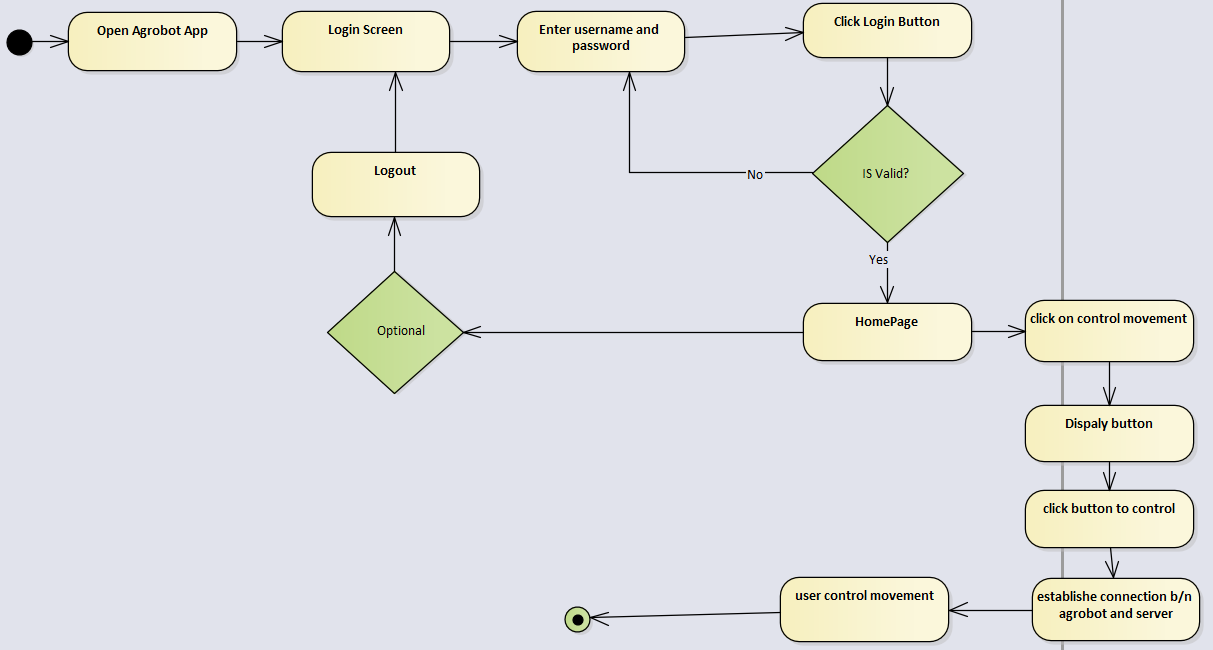


Figure 22 activity diagram user control movement

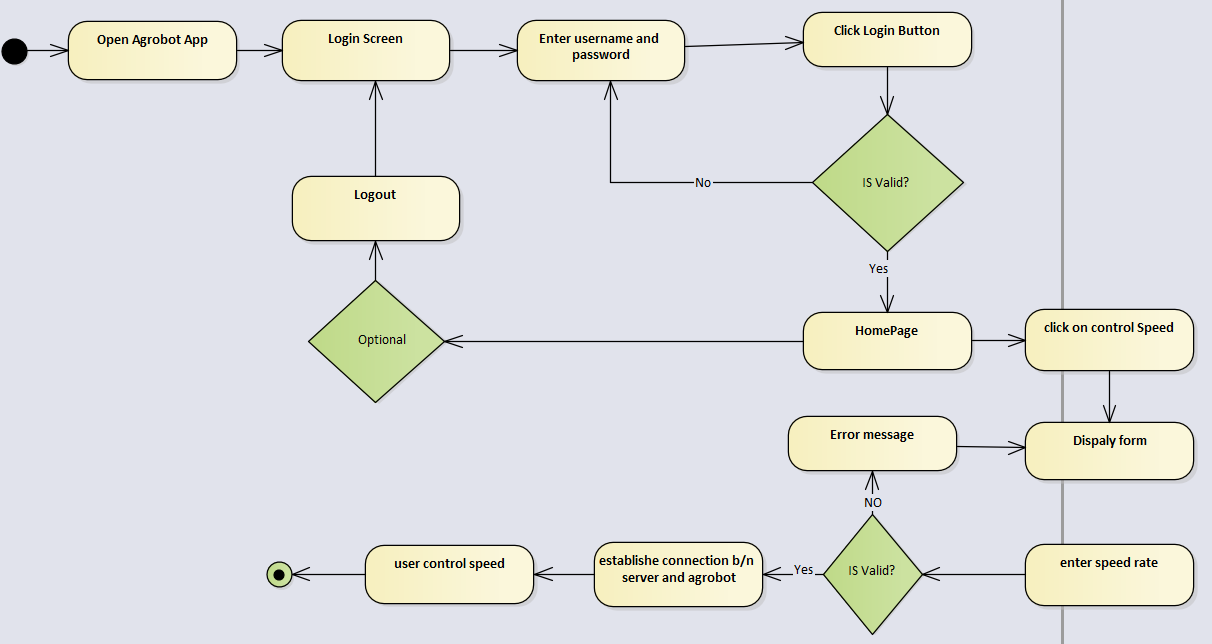
**

Figure 23 activity diagram user control speed

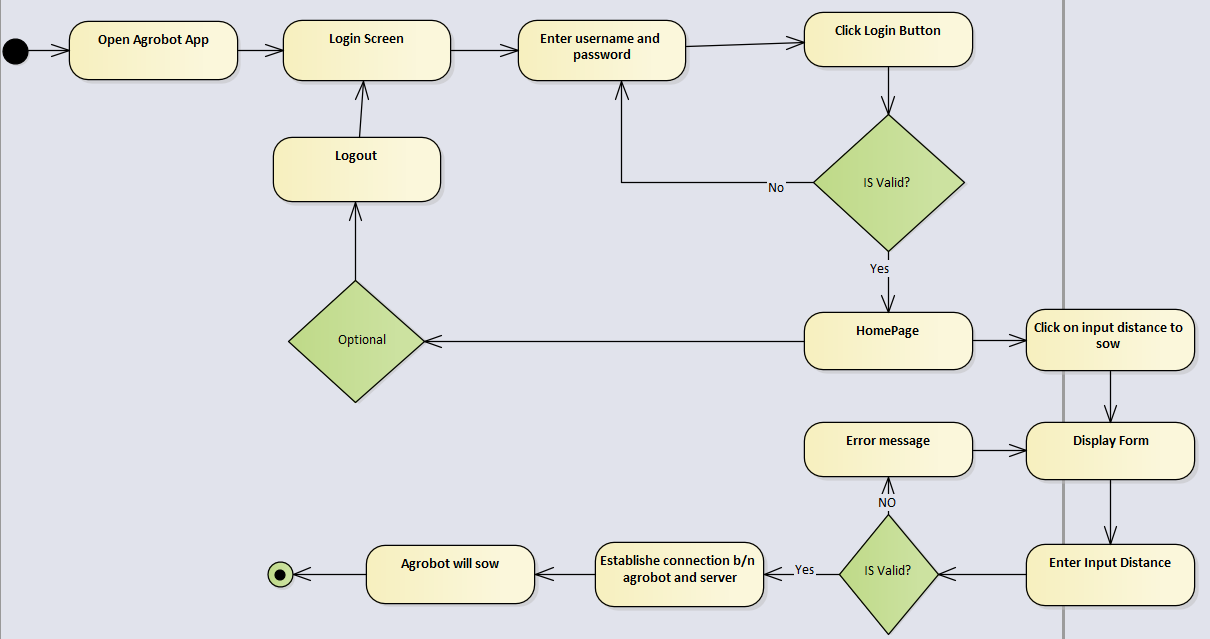


Figure 24 activity diagram user input distance to sow

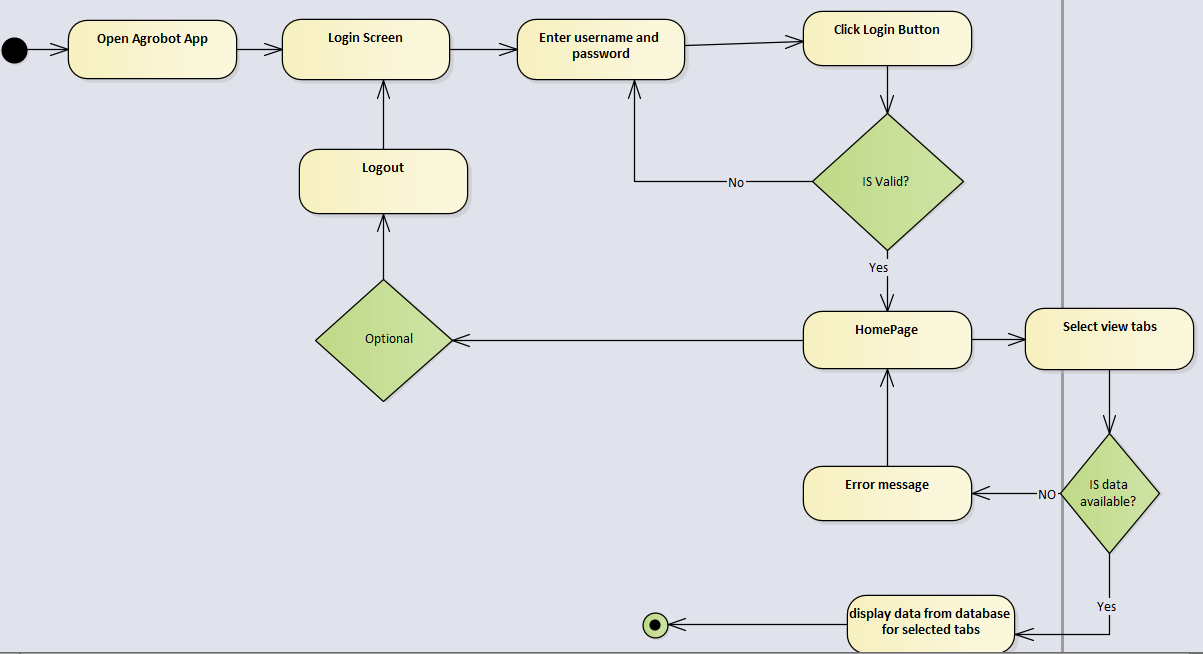


Figure 25 activity diagram user can view details

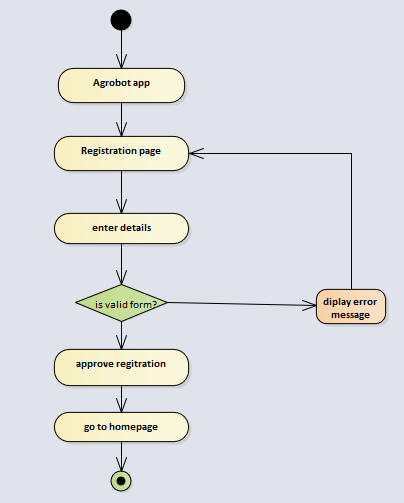


Figure 26 Activity diagram for registration

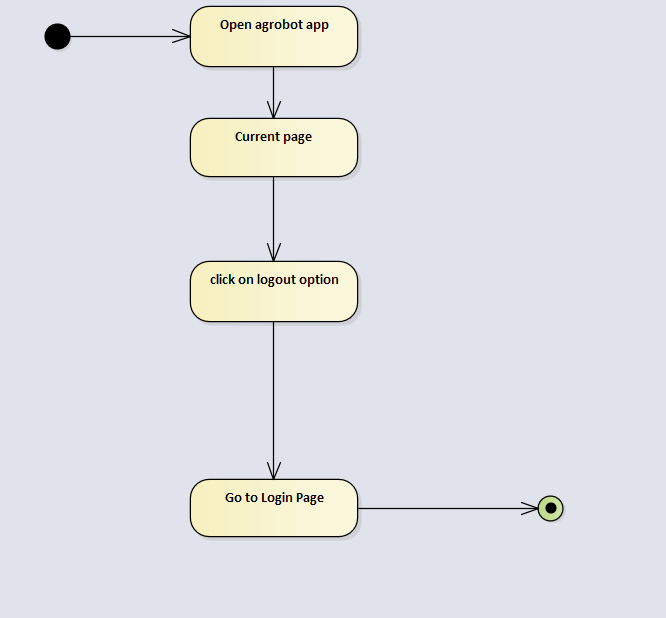
**

Figure 27 activity diagram for User can logout

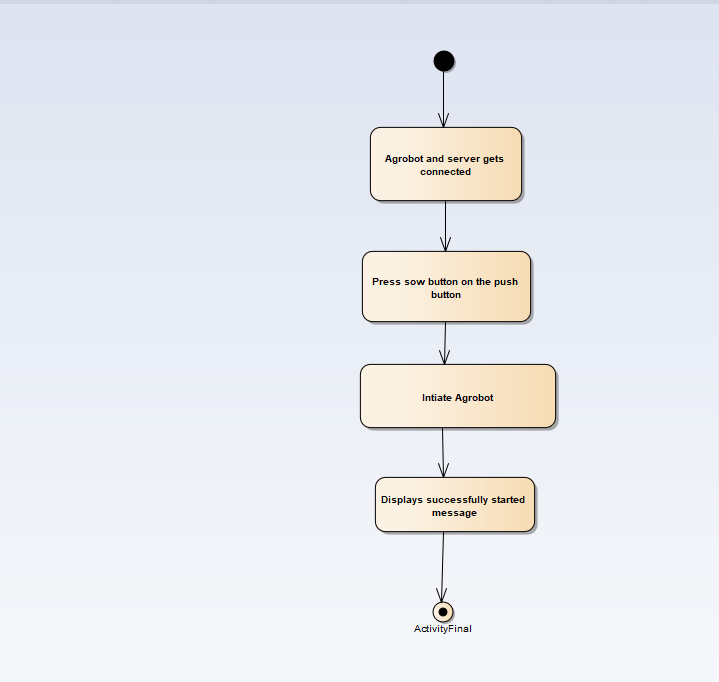


Figure 28 activity diagram for agrobot can sow

## **State Diagram**

State diagrams are used to give an abstract description of the [behavior](http://en.wikipedia.org/wiki/Behavior) of a [system.](http://en.wikipedia.org/wiki/System) This behavior is analyzed and represented in series of events that could occur in one or more possible states.

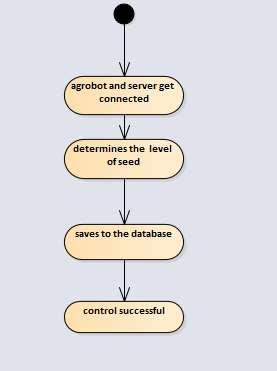
**.**

Figure 29 state chart diagram Agrobot control movement

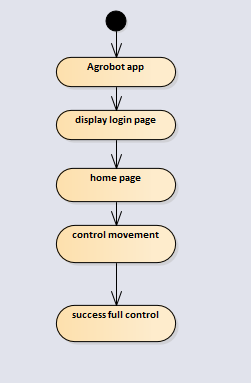
****

Figure 30 state chart diagram user control movement

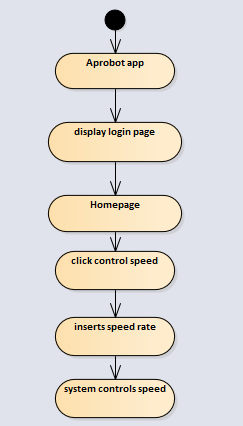
****

Figure 31 state chart diagram user control Speed

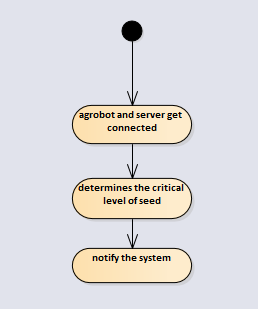
****

Figure 32 state chart diagram Agrobot determine the critical seed level

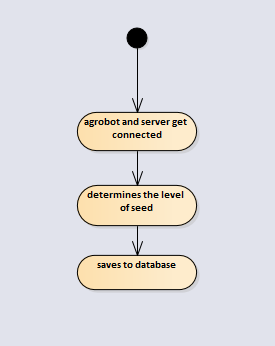
****

Figure 33 state chart diagram Agrobot determine seed level

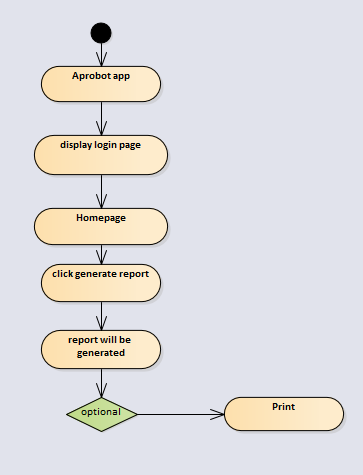
****

Figure 34 state chart diagram user generate report

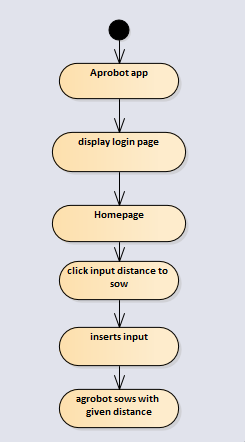
****

Figure 35 state chart diagram user enters input distance to sow

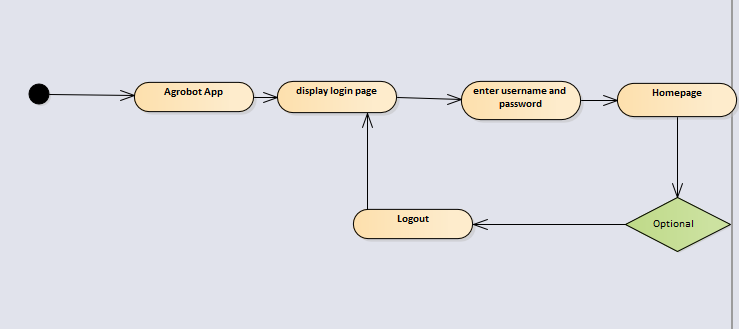
****

Figure 36 state chart diagram user login to Agrobot

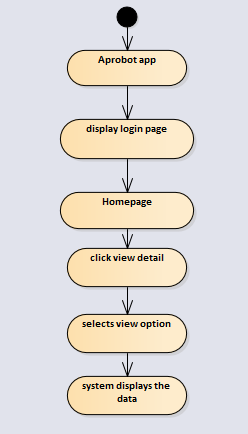
****

Figure 37 state chart diagram user can view details

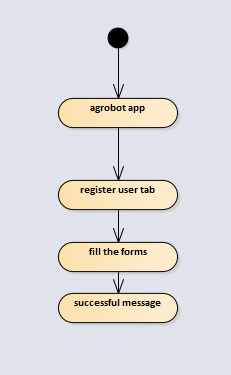
****

Figure 38 state chart diagram of registration

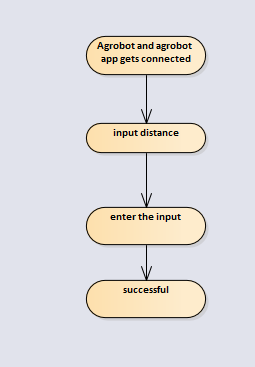
****

Figure 39 state chart diagram Agrobot input distance to sow

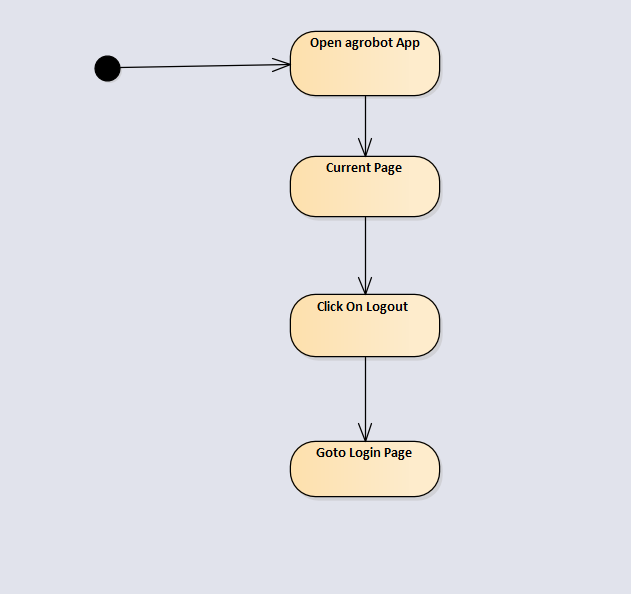


Figure 40 state chart diagram for User can logout

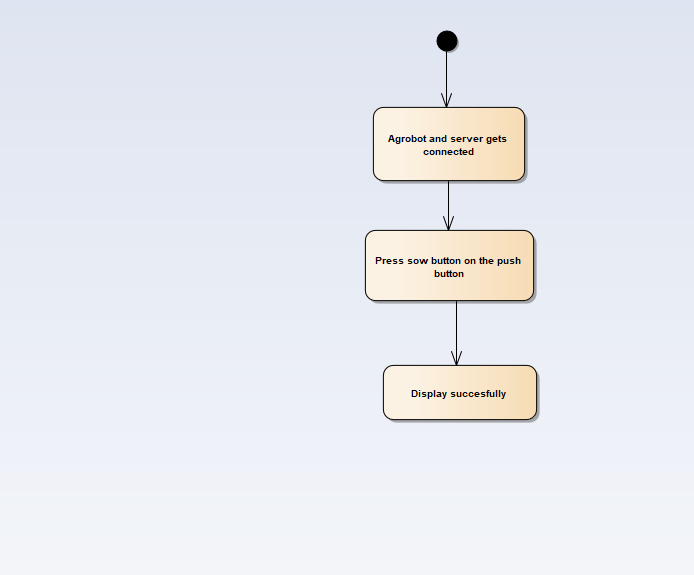


Figure 41 State chart diagram for User can sow

# CHAPTER FOUR

# SYSTEM DESIGN

## Overview

**System design** is the process of designing the elements of a system such as the architecture, modules and components, the different interfaces of those components and the data that goes through that system. This is the system design document for *Teff* sowing Agrobot which will operate on farmland used to simplify manual row plant mechanism. This document includes proposed system design, design goal and object design.

### Purpose of the System Design

Design is mainly used to show the direction how the system is built and to obtain clear and enough information needed to drive the actual implementation of the system. This chapter describes the design issues of the overall system. This provides the complete architectural overview of the proposed system. The designer should know that the nature of design greatly affects its ease of implementation, which determines its quality. Once after the system is implemented and if one needs to manipulate some part of system, it will greatly depend on our quality of system design.

### Design Goal

Eventually goal of design is used to model the system with high quality. This will be used by developers to describe the qualities of the system that another developer should consider. These design goals represent the desired qualities that the system should have and provide a consistent set of criteria that would be taken into consideration when making design decisions. Description of feature character and attribute of the system can be found from non-functional requirement. These nonfunctional requirements can be:

* Performance
* Dependability
* Maintenance
* End User

**Performance**

Agrobot as well as Agrobot software respond fast should perform the task quickly Possible as possible. Establishing communication, data sending and receiving, execution of received data and also communication of serial data, communication between sensors is very fast. Agrobot software performs its task such as generating report and receiving, viewing robot status within a user acceptable time and space. This includes the following:

* Response time: - depending on the bound rate of Agrobot and available network, the system will response in short period of time.
* Storage space: - to do work efficiently Agrobot software has more than 50% RAM and Agrobot software is install on the processor more than 2GB RAM

**Dependability**

Our system includes the following dependability criteria’s: -

* Reliability: Agrobot and Agrobot software should be reliable.
* Fault Tolerance: - Agrobot and Agrobot software should be fault tolerant due to termination of communication.
* Security: -Agrobot software should be secured, i.e., unauthorized user cannot manipulate Agrobot through Agrobot software
* Availability: - Agrobot can operate with any weather condition as well as day and night, also as long as there is local area network and establish communication Agrobot software can operate and it will be available 24 hours a day.

**Maintainability**

To be maintainable the system should meet the following maintenance criteria: -

* Modifiability: Agrobot is modifiable for further modification just preparing new embedded code as well as hardware change. Also, Agrobot software is modifiable like version control.
* Portability: - Agrobot software is developed for android mobile.
* Extensibility: - if it is needed to add new functionality to Agrobot, new circuit diagram and embedded code needed for Agrobot. New functionality to Agrobot software can be achieved by only making a separate page and integrate this page with the existing system.

**End User Criteria**

* Agrobot software should be simple and user friendly. It has understandable graphical user interface such as forms and buttons, which have descriptive names. All the interfaces, forms and buttons are written or designed in a simple language or common language (English, Afaan Oromoo, and Amharic) so that the user can access it without any difficulty. Also, Agrobot also have descriptive pushbutton.

## **Proposed system**

### **Overview**

The proposed system is mainly based on the SRS document that’s already submitted. It mainly deals with subsystem decomposition – dividing the system in to manageable components. Another major task in system design deals with hardware/software mapping which deals with which components would be part in which hardware. Yet another aspect of system design deals with persistent data management, which illustrate as to how persistent data (file, database, etc.) are stored and managed and at last Access control will be presented.

Our system is three-tier architecture and it has client side, server and database.

**Client side**: here in the client-side user, Agrobot, and user interface will be existing

**Server side**: web servers to connect the data base application are found; mean that the  
application logic to perform the application by the client is found.  
**Data base:** here the data bases that store the information are found.

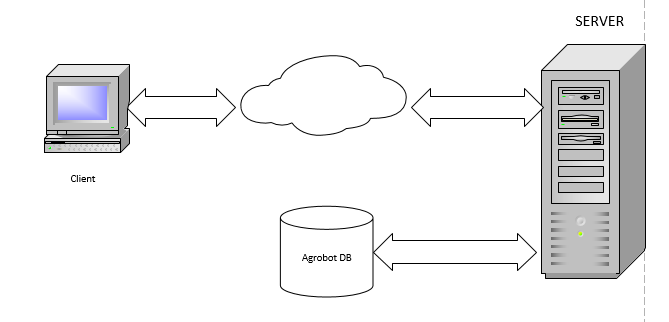


Figure 42 system architecture of TSA

### **System process**

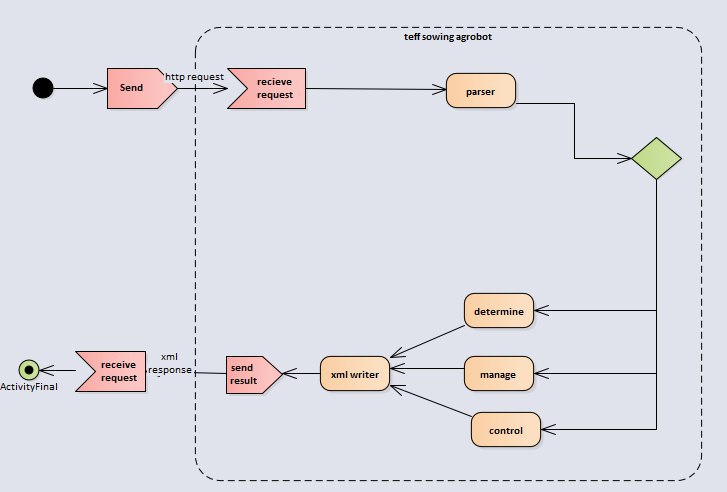
****

Figure 43 Agrobot system process TSA

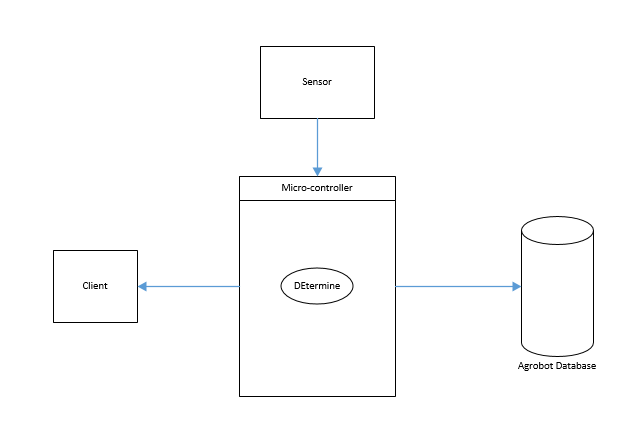


Figure 44 Agrobot determine system overview

The above figure 47 indicates Microcontroller controls and coordinates the sensors properly. The microcontroller performs tasks based on the sensing data from the sensors like determine the level of seeds, position of robot, determine the critical seed level and the farmland coverage, then the necessary data will save to database. When microcontroller determines the critical seed level it alarms the user.

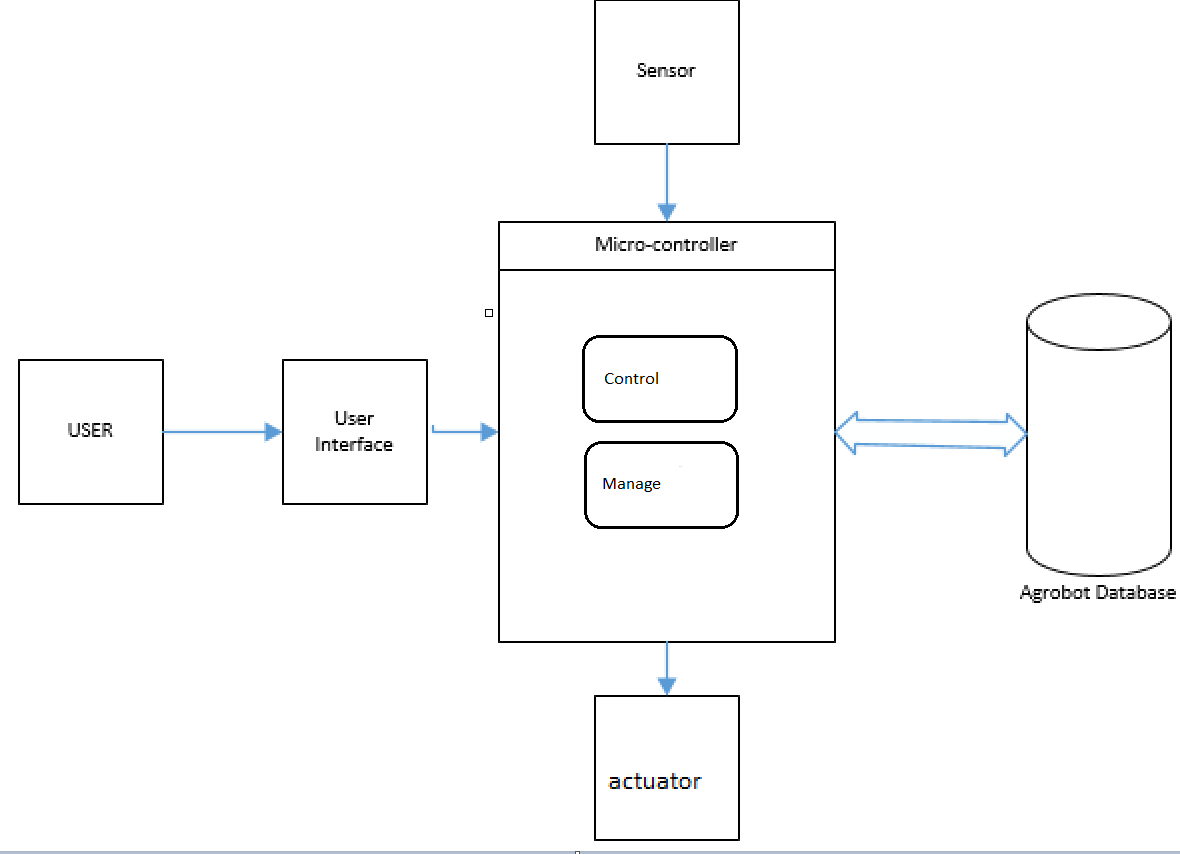


Figure 45 Agrobot control and manage input from user system overview

The above figure 48 indicates that the user give input to microcontroller using keypad or button or other input mechanisms and the microcontroller take sensing data from different sensors and microcontroller will do the speed and movement in controller module ,input distance to sow and row spacing in manage modules based on user inputs and coordinate the sensors and the microcontroller controller the mechanical parts by sending data to the actuators then the necessary information will save to database.

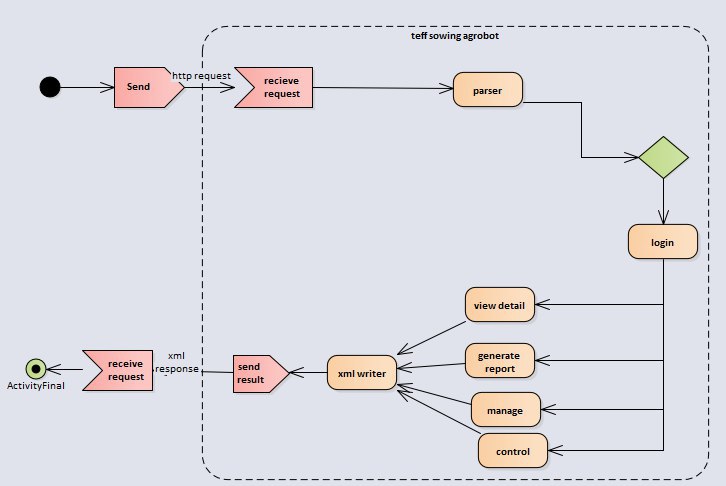
****

Figure 46 Agrobot software system process of TSA

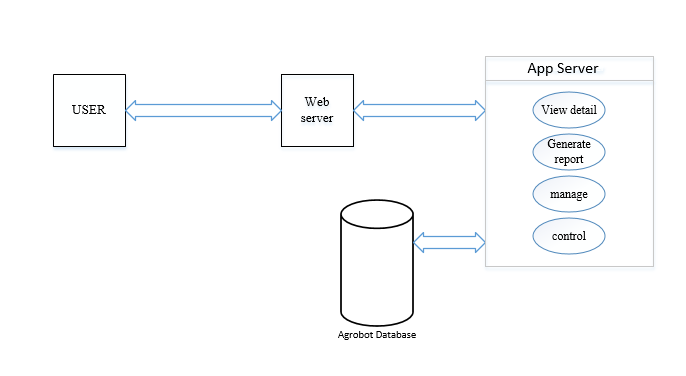


Figure 47 Agrobot software side system over view

### **Subsystem decomposition**

To reduce the complexity of the solution domain, we decompose a system into simpler parts, called subsystems. The main need of this portion is to design the internal part of the system. In this project, there are three sub system decompositions.

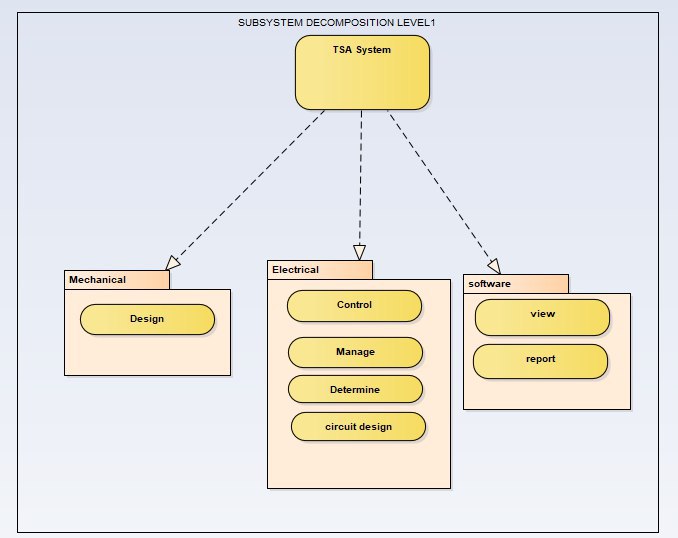


Figure 48 subsystem decomposition of TSA level 1

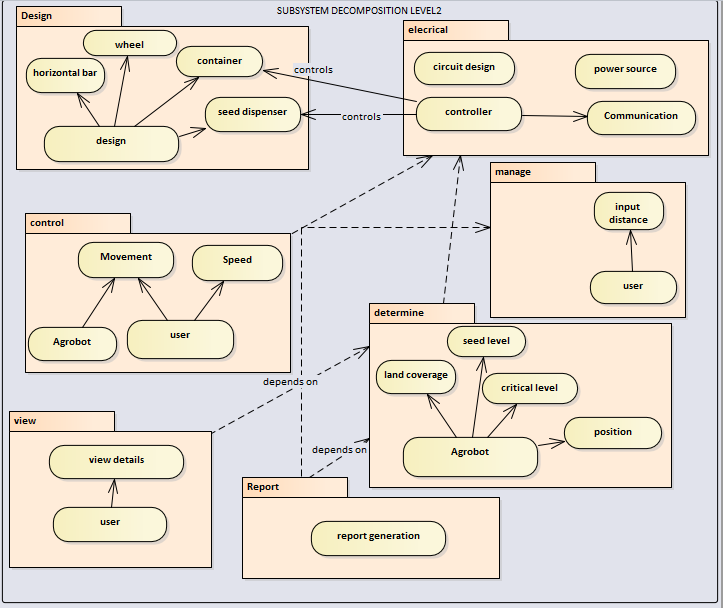


Figure 49 subsystem decomposition of TSA level 2

**Subsystem decomposition description**

Table 25 subsystem description

|  |  |  |
| --- | --- | --- |
| Subsystem | Purpose | Class |
| Software | Is responsible for viewing details, managing and generating report. | * User * sensor value * report |
| Electrical | Is responsible for controlling speed, movement, communication between the components, manage input distance and row spacing, determine (position, level of seed, critical level, farmland coverage). | * User * Agrobot * sensor value |
| Mechanical | Is responsible for designing of the mechanical parts of the Agrobot (seed dispenser, horizontal bar, seed container, wheel, structure) |  |

### **Hardware/software mapping**

When we say hardware/software mapping for the system, it describes how subsystems are assigned to hardware and off-the-shelf components. It also lists the issues introduced by multiple nodes and software reuse. In this system design mainly, there are three hardware components. The client side (user, Agrobot), server side and database side.



Figure 50 Hardware/software mapping of TSA for Agrobot software side

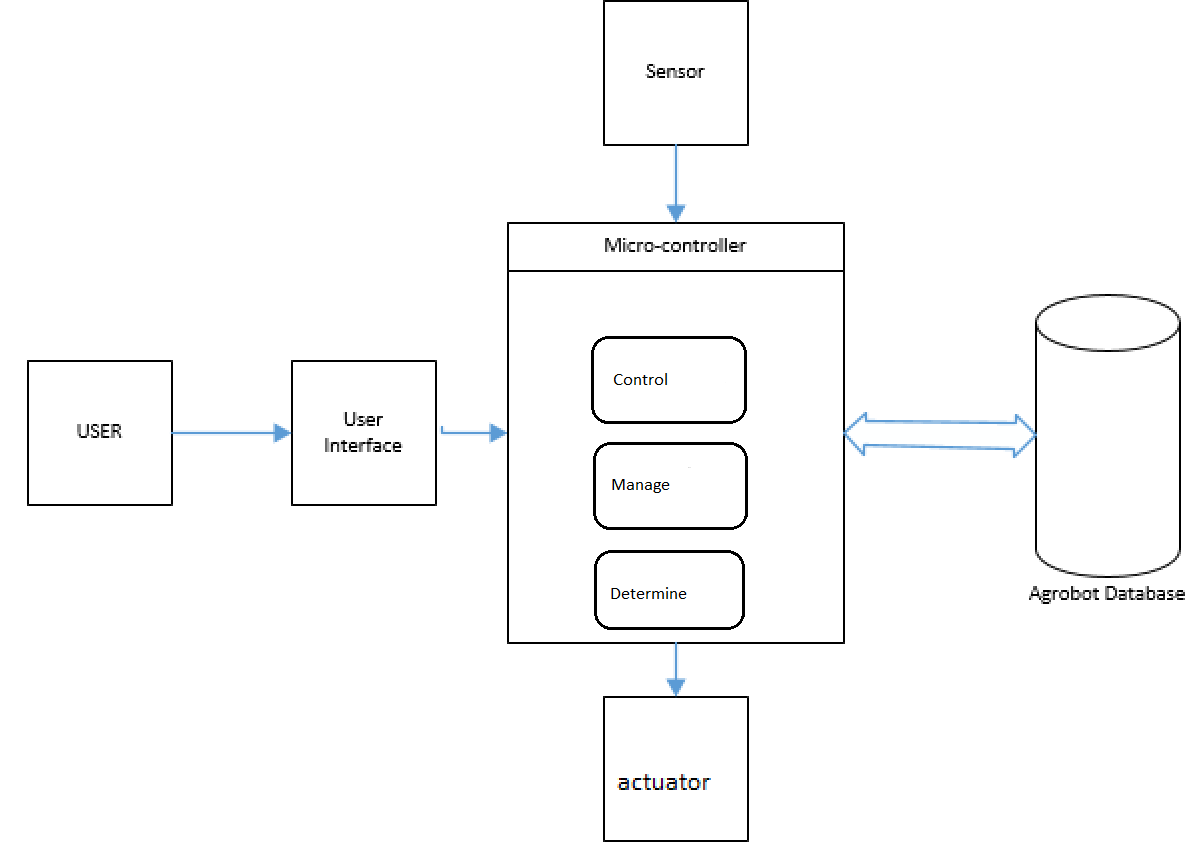
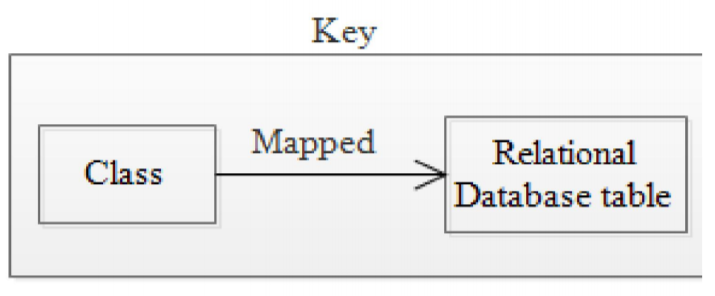


Figure 51 Hardware/software mapping of TSA for Agrobot

The above figure 54 indicates that the user give input to microcontroller using keypad or button or other input mechanisms and the microcontroller take sensing data from different sensors and microcontroller will do the speed and movement controller, determine and input distance to sow and row spacing manage modules based on user inputs and coordinate the sensors and the microcontroller controller the mechanical parts by sending data to the actuators then the necessary information will save to database.

### **Persistent data management**

The purpose of this section is to show the mapping of the objects/classes of the system, identified during the analysis stage, in to the corresponding relational database.



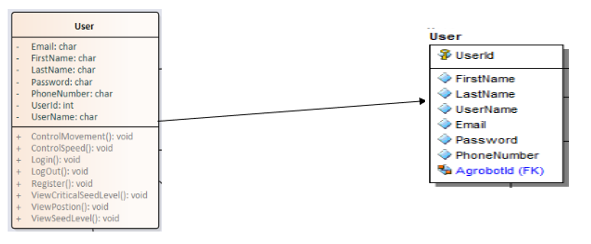


Figure 52 object for user

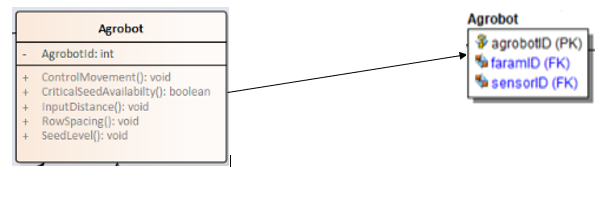


Figure 53 object for Agrobot

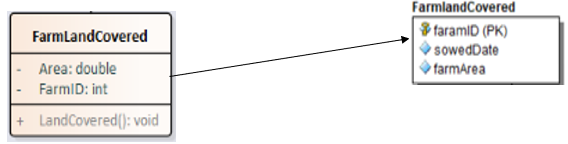


Figure 54 object for farmland covered

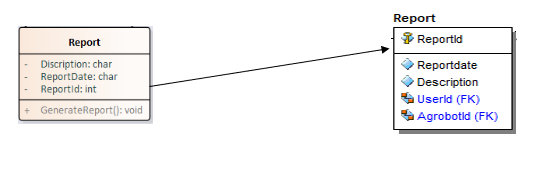


Figure 55 object for report

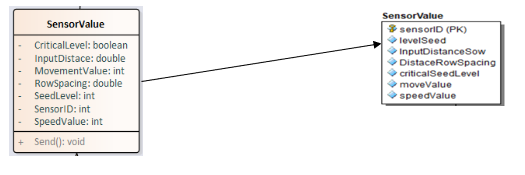


Figure 56 object for sensor value



Figure 57 object for reset password

### **Component diagram**

The following component diagram represents a group of graphs of components connected by dependency relationships and dependencies are shown as dashed arrows from the client component to the supplier component.

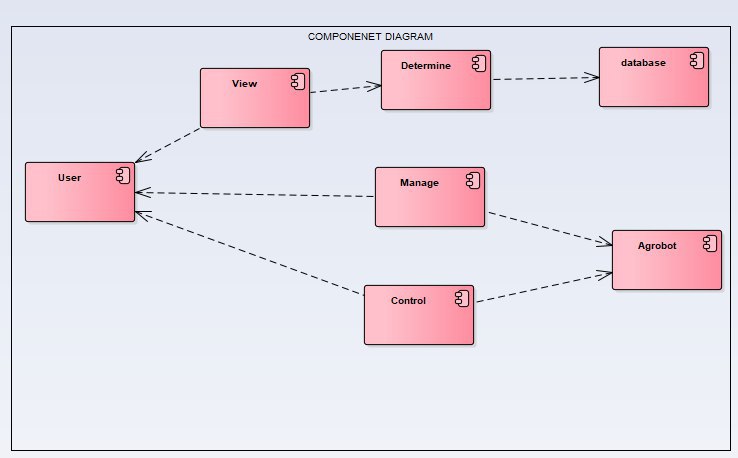


Figure 58 Component diagram of TSA

### **Deployment diagram**

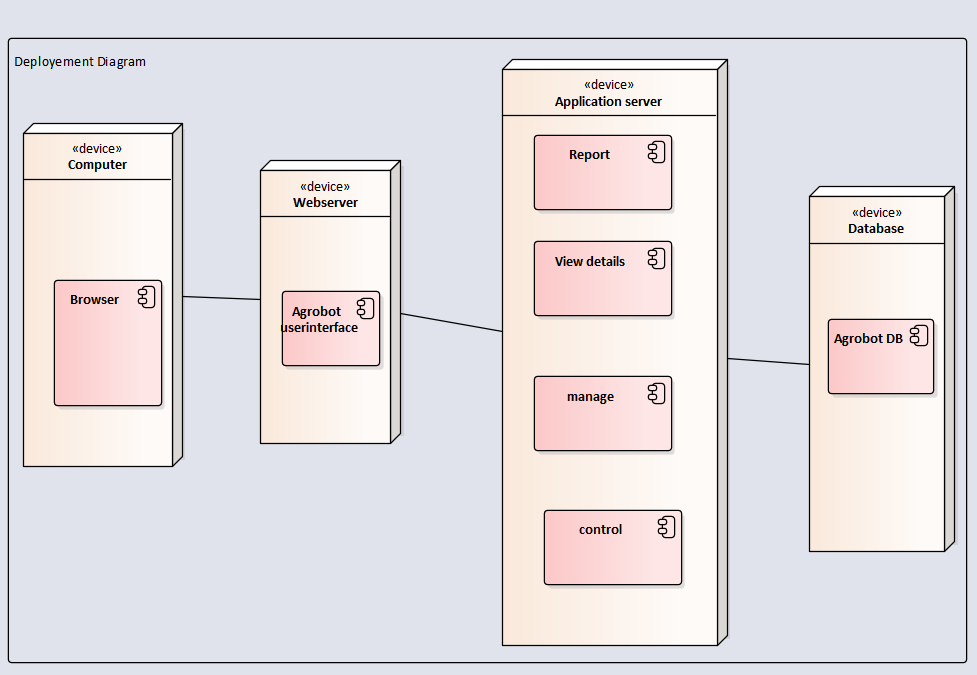
****

Figure 59 deployment diagram of TSA for Agrobot software side

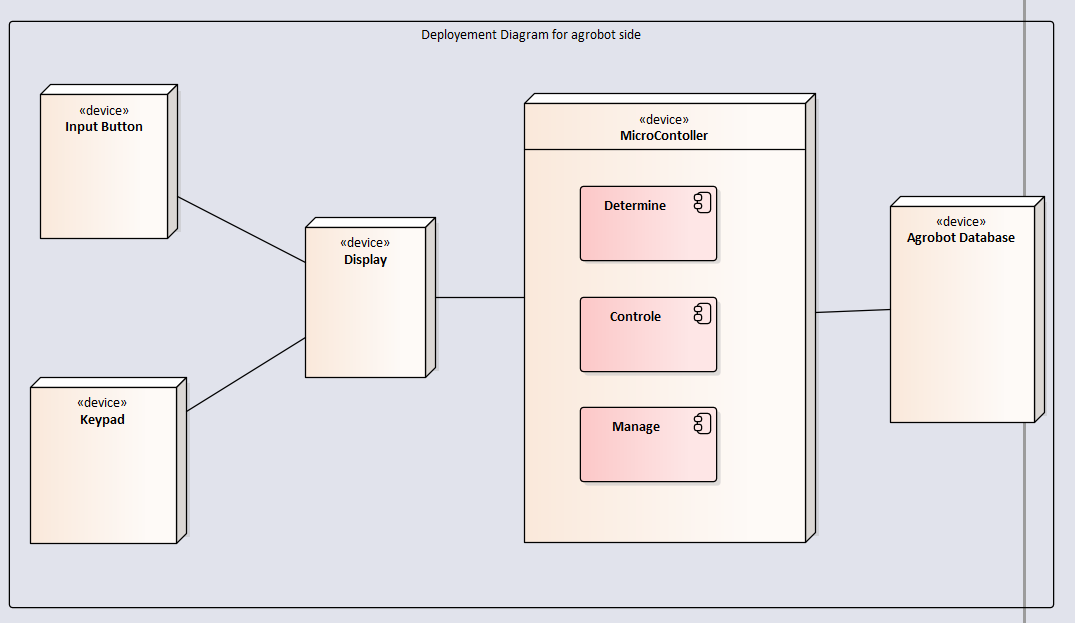
****

Figure 60 deployment diagram of TSA for Agrobot side

### **Database design**

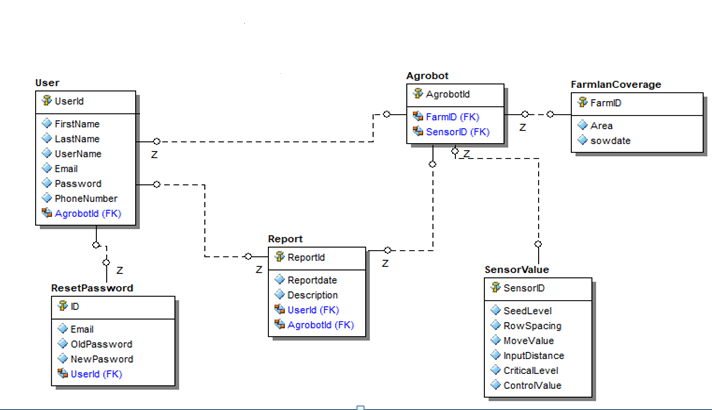
****

Figure 61 database design of TSA

### **Access control**

In multiuser systems, different actors have access to different functionality and data. We modeled these distinctions by associating different use cases to different actors. This is described by the following access matrix.

*Table 26 access control*

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Object actor | Viewdetails | Control | Determine | Manage | Report | Account | Access |
| User | Viewseedlevel()  Viewposition()  Viewcriticalseedlevel() | Speed()  Movement() | **-** | Rowspacing()  Inputdistance() | Generatereport() | Createaccount() | Logout()  Login()  Reset() |
| Agrobot |  | Movement() | Seedlevel()  Position()  Criticalseedlevel() | Inputdistance() |  |  |  |

### **User interface design**

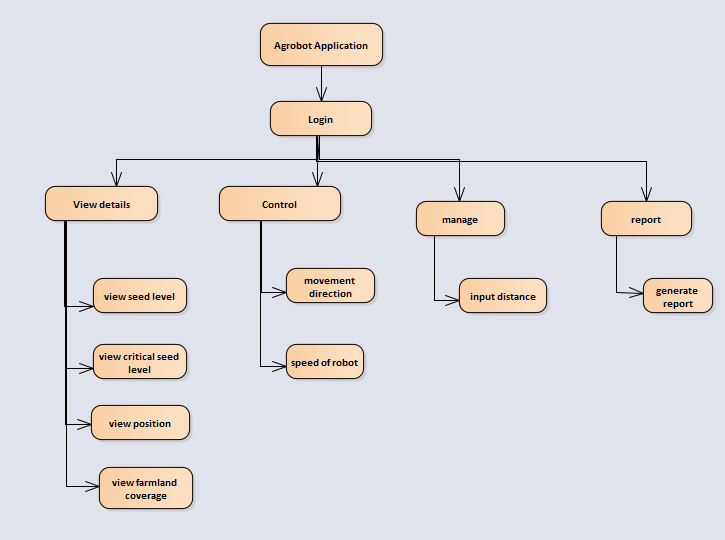
****

Figure 62 user interface design of TSA for Agrobot software side

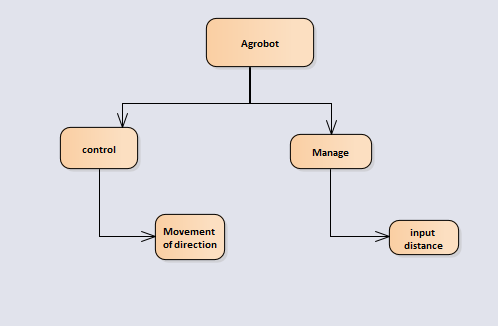
**

Figure 63 user interface design of TSA for Agrobot side

# Chapter Five

# Implementation

## Overview

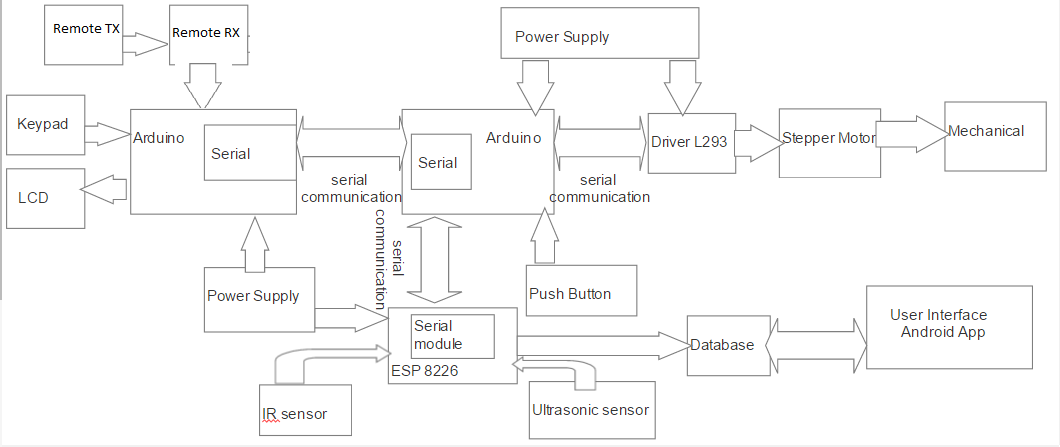
Implementation is the process of realizing the design as a program.

This section of the Project Implementation Part describes the support hardware, software, facilities, and materials required for the implementation, as well as the documentation, necessary personnel and training requirements, outstanding issues and implementation impacts to the current environment.

## Tools and technologies utilized during system development

We used different tools and technologies to develop our system. These tools include hardware and software tools for both client side and server side to develop the system.

**System Block Diagram**



`

Figure 64 System block diagram

**Hardware tools**

**Microcontroller**

Arduino is an open source hardware/software programming platform based around Atmel microcontrollers. Arduino is an open-source platform used for building electronics projects. Arduino consists of both a physical programmable circuit board (often referred to as a microcontroller) and a piece of software, or IDE (Integrated Development Environment) that runs on your computer, used to write and upload computer code to the physical board. The Arduino platform has become quite popular with people just starting out with electronics, with ATMEGA 328 board. Unlike most previous programmable circuit boards, the Arduino UNO does not need a separate piece of hardware (called a programmer) in order to load new code onto the board, It simply use a USB cable. Additionally, the Arduino IDE uses a simplified version of C languages, making it easier to learn to program. Finally, Arduino provides a standard form factor that breaks out the functions of the micro-controller into a more accessible package [10].

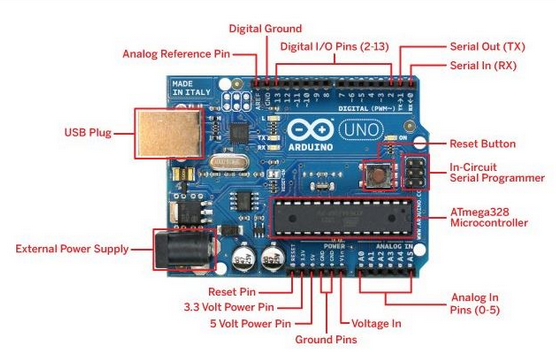


Figure 65 Arduino-Uno and pin configurations

**Power (USB) and pin configuration**

Every Arduino board needs a way to be connected to a power source. The Arduino UNO can be powered from a USB cable coming from your computer or a wall power supplies. It’s not allowed to use a power supply greater than 20 Volts as you will overpower and thereby destroy the Arduino UNO. The recommended voltage for most Arduino models is between 6 and 12 Volts. The pins on your Arduino are the places where it will connect wires to construct a circuit probably in conjunction with a breadboard and some wire. They usually have black plastic „headers‟ that allow you to just plug a wire right into the board. The Arduino has several different kinds of pins, each of which is labeled on the board and used for different functions.

* **GND (3):** Short for „Ground‟. There are several GND pins on the Arduino, any of which can be used to ground the circuit.
* **5V (4) & 3.3V (5):** As we might guess, the 5V pin supplies 5 volts of power, and the 3.3V pin supplies 3.3 volts of power. Most of the simple components used with the Arduino run happily off of 5 or 3.3 volts.
* **Analog (6):** The area of pins under the „Analog In‟ label (A0 through A5 on the UNO) are Analog In pins. These pins can read the signal from an analog sensor (like a temperature sensor) and convert it into a digital value that we can read.
* **Digital (7):** Across from the analog pins are the digital pins (0 through 13 on the UNO). These pins can be used for both digital input (like telling if a button is pushed) and digital output (like powering an LED).
* **PWM (8):** the digital pins (3, 5, 6, 9, 10, and 11 on the UNO). These pins act as normal digital pins, but it can also be used for something called Pulse-Width Modulation (PWM).
* **AREF (Stands for Analog Reference) (9):** Most of the time you can leave this pin alone. It is sometimes used to set an external reference voltage (between 0 and 5 Volts) as the upper limit for the analog input pins [11].

Table 27 Pin Description

|  |  |  |
| --- | --- | --- |
| Pin Category | Pin Name | Details |
| Power | Vin, 3.3V, 5V, GND | Vin: Input voltage to Arduino when using an external power source.  5V: Regulated power supply used to power microcontroller and other components on the board 3.3V: 3.3V supply generated by on-board voltage regulator. Maximum current draw is 50mA.GND: ground pins. |
| Reset | Reset | Resets the microcontroller. |
| Analog Pins | A0 – A5 | Used to provide analog input in the range of 0-5V |
| Input/output Pins | Digital Pins 0 - 13 | Can be used as input or output pins. |
| Serial | 0(Rx), 1(Tx) | Used to receive and transmit TTL serial data. |
| External Interrupts | 2, 3 | To trigger an interrupt. |
| PWM | 3, 5, 6, 9, 11 | Provides 8-bit PWM output. |
| SPI | 10(SS),11 (MOSI), 12 (MISO) and 13 (SCK) | Used for SPI communication. |
| Inbuilt LED | 13 | To turn on the inbuilt LED. |
| TWI | A4(SDA), A5 (SCA) | Used for TWI communication. |
| AREF | AREF | To provide reference voltage for input voltage. |

Table 28 Arduino Uno Technical Specifications

|  |  |
| --- | --- |
| Microcontroller | ATmega328P – 8 bit AVR family microcontroller |
| Operating Voltage | 5V |
| Recommended Input Voltage | 7-12V |
| Input Voltage Limits | 6-20V |
| Analog Input Pins | 6 (A0 – A5) |
| Digital I/O Pins | 14 (Out of which 6 provide PWM output) |
| DC Current on I/O Pins | 40 mA |
| DC Current on 3.3V Pin | 50 mA |
| Flash Memory | 32 KB (0.5 KB is used for Bootloader) |
| SRAM | 2 KB |
| EEPROM | 1 KB |
| Frequency (Clock Speed) | 16 MHz |

**Stepper Speed Control**

A Stepper Motor or a step motor is a brush less, synchronous motor, which divides a full rotation into a number of steps. Unlike a brush less DC motor, which rotates continuously when a fixed DC voltage is applied to it, a step motor rotates in discrete step angles [12].

Stepper motors, due to their unique design, can be controlled to a high degree of accuracy without any feedback mechanisms. The shaft of a stepper, mounted with a series of magnets, is controlled by a series of electromagnetic coils that are charged positively and negatively in a specific sequence, precisely moving it forward or backward in small "steps".

There are two types of steppers, Unipolar and Bipolar, and it is very important to know which type you are working with. For each of the motors, there is a different circuit. The example code will control both kinds of motors. See the unipolar and bipolar motor schematics for information on how to wire up your motor [13].

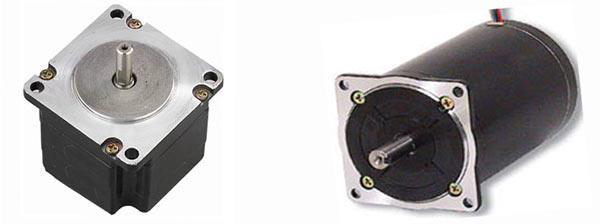


Figure 66 Stepper motor

How a Stepper Motor Works?

A regular DC motor spins in only direction whereas a Stepper motor can spin in precise increments.

Stepper motors can turn an exact amount of degrees (or steps) as desired. This gives you total control over the motor, allowing you to move it to an exact location and hold that position. It does so by powering the coils inside the motor for very short periods of time. The disadvantage is that you have to power the motor all the time to keep it in the position that you desire.

All you need to know for now is that, to move a stepper motor, you tell it to move a certain number of steps in one direction or the other, and tell it the speed at which to step in that direction. There are numerous varieties of stepper motors. The methods described here can be used to infer how to use other motors and drivers which are not mentioned in this tutorial. However, it is always recommended that you consult the datasheets and guides of the motors and drivers specific to the models you have.

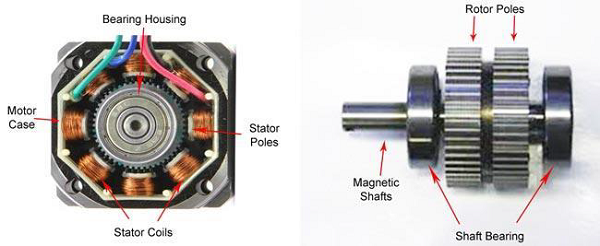


Figure 67 Interior of stepper motor

**LCD** (**Liquid Crystal** **Display)** screen is an electronic display module and find a wide range of applications. A **16x2 LCD** display is very basic module and is very commonly used in various devices and circuits. ... A **16x2 LCD** means it can display 16 characters per line and there are 2 such lines. The pin **7** contains total **8 pins** which are used as a Data pin of LCD. This pin is connected to VCC and it is used for the pin **16** to set up the glow of backlight of LCD [14].



Figure 68 16×2 ***LCD***

The LCDs have a parallel interface, meaning that the micro-controller has to manipulate several interface pins at once to control the display. The interface consists of the following pins:

* A register selects (RS) pin that controls where in the LCD's memory you're writing data to. You can select either the data register, which holds what goes on the screen, or an instruction register, which is where the LCD's controller looks for instructions on what to do next.
* A Read/Write (R/W) pin that selects reading mode or writing mode
* An Enable pin that enables writing to the registers
* 8 data pins (D0 -D7). The states of these pins (high or low) are the bits that you're writing to a register when you write, or the values you're reading when you read.

There's also a display contrast pin (Vo), power supply pins (+5V and Gnd) and LED Backlight (Bklt+ and BKlt-) pins that you can use to power the LCD, control the display contrast, and turn on and off the LED backlight, respectively.

The process of controlling the display involves putting the data that form the image of what you want to display into the data registers, then putting instructions in the instruction register. The [Liquid Crystal Library](https://www.arduino.cc/en/Reference/LiquidCrystal) simplifies this for you so you don't need to know the low-level instructions.

The Hitachi-compatible LCDs can be controlled in two modes: 4-bit or 8-bit. The 4-bit mode requires seven I/O pins from the Arduino, while the 8-bit mode requires 11 pins. For displaying text on the screen, you can do most everything in 4-bit mode, so example shows how to control a 16x2 LCD in 4-bit mode.

**Features**

* LCD (liquid crystal display) is the technology used for displays in notebook and other smaller computers. Like light-emitting diode (LED) and gas-plasma technologies, LCDs allow displays to be much thinner than cathode ray tube (CRT) technology.
* LCDs consume much less power than LED and gas-display displays because they work on the principle of blocking LCD (liquid crystal display) is the technology used for displays in notebook and other smaller computers. Like light-emitting diode (LED) and gas-plasma technologies, LCDs allow displays to be much thinner than cathode ray tube (CRT) technology.

**Keypad**

A keypad is one of the most commonly used input devices in microprocessor applications. In a standard keypad wired as an X-Y switch matrix, normally-open switches connect a row to a column when pressed. If a keypad has 12 keys, it is wired as 3 columns by 4 rows [15].



Figure 69 Keypad

Keypads are a great way to let users interact with the project. We use them to navigate menus, enter passwords, and control games and robots.

We include a 7-pin extra-long header strip so you can plug this into a breadboard with ease.

**Features:**

* Ultra-thin design & adhesive backing provides easy integration to any project
* Easy communication with any micro-controller
* Maximum Rating: 24 VDC, 30 mA
* Interface: 7-pin access to 3x4 matrix

**Breadboard**

A breadboard is a solder less device for temporary prototype with electronics and test circuit designs. Most electronic components in electronic circuits can be interconnected by inserting their leads or terminals into the holes and then making connections through wires where appropriate. The breadboard has strips of metal underneath the board and connects the holes on the top of the board. The metal strips are laid out as shown below. Note that the top and bottom rows of holes are connected horizontally and split in the middle while the remaining holes are connected vertically [16].

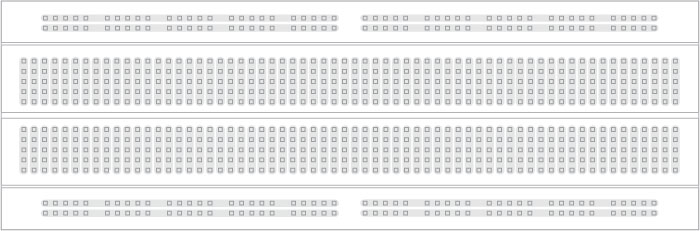


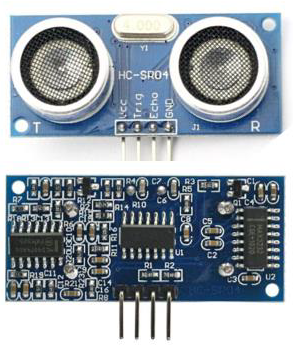
Figure 70 Breadboard

**Ultrasonic Sensor**

The HC-SR04 ultrasonic sensor uses SONAR to determine the distance of an object just like the bats do. It offers excellent non-contact range detection with high accuracy and stable readings in an easy-to-use package from 2 cm to 400 cm or 1” to 13 feet.

The operation is not affected by sunlight or black material, although acoustically, soft materials like cloth can be difficult to detect. It comes complete with ultrasonic transmitter and receiver module [17].

.



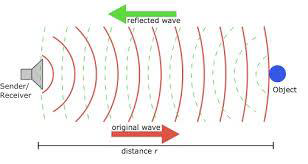


Figure 71 Ultrasonic sensor

Technical Specifications

* Power Supply − +5V DC
* Quiescent Current − <2mA
* Working Current − 15mA
* Effectual Angle − <15°
* Ranging Distance − 2cm – 400 cm/1″ – 13ft
* Resolution − 0.3 cm
* Measuring Angle − 30 degree

**ESP NodeMCU Wi-Fi Module**

NodeMCU is an open source IoT platform. The **ESP8266 WiFi Module** is a self-contained SOC with integrated TCP/IP protocol stack that can give any micro-controller access to your Wi-Fi network. The ESP8266 is capable of either hosting an application or offloading all Wi-Fi networking function from another application processor [18].



Figure 72 **ESP NodeMCU**

**How** **ESP NodeMCU Wi-Fi Module works**

**Features**

* Processor: L106 32-bit [RISC](https://en.wikipedia.org/wiki/Reduced_instruction_set_computing) microprocessor core based on the [Tensilica](https://en.wikipedia.org/wiki/Tensilica) Xtensa Diamond Standard 106Micro running at 80 MHz
* Memory:
  + 32 KiB instruction RAM
  + 32 KiB instruction cache RAM
  + 80 KiB user-data RAM
  + 16 KiB ETS system-data RAM
* External QSPI flash: up to 16 MiB is supported (512 KiB to 4 MiB typically included)
* IEEE 802.11 b/g/n Wi-Fi
  + Integrated TR switch, [balun](https://en.wikipedia.org/wiki/Balun), [LNA](https://en.wikipedia.org/wiki/Low-noise_amplifier), [power amplifier](https://en.wikipedia.org/wiki/RF_power_amplifier) and [matching network](https://en.wikipedia.org/wiki/Matching_network)
  + [WEP](https://en.wikipedia.org/wiki/Wired_Equivalent_Privacy) or [WPA/WPA2](https://en.wikipedia.org/wiki/Wi-Fi_Protected_Access) authentication, or open networks
* 16 [GPIO](https://en.wikipedia.org/wiki/General-purpose_input/output) pins
* [SPI](https://en.wikipedia.org/wiki/Serial_Peripheral_Interface_Bus)
* [I²C](https://en.wikipedia.org/wiki/I%C2%B2C) (software implementation)
* [I²S](https://en.wikipedia.org/wiki/I%C2%B2S) interfaces with DMA (sharing pins with GPIO)
* [UART](https://en.wikipedia.org/wiki/Universal_asynchronous_receiver/transmitter) on dedicated pins, plus a transmit-only UART can be enabled on GPIO2
* 10-bit [ADC](https://en.wikipedia.org/wiki/Analog-to-digital_converter) ([successive approximation ADC](https://en.wikipedia.org/wiki/Successive_approximation_ADC))

**Pushbutton**

Push buttons or switches connect two points in a circuit when you press them. This example turns on one led when the button pressed once, and off when pressed twice

We connect three wires to the Arduino board. The first goes from one leg of the push button through a pull-up resistor (here 2.2 KOhms) to the 5-volt supply. The second goes from the corresponding leg of the push button to ground. The third connects to a digital i/o pin (here pin 7) which reads the button's state [19].

When the push button is open (unpressed) there is no connection between the two legs of the push button, so the pin is connected to 5 volts (through the pull-up resistor) and we read a HIGH. When the button is closed (pressed), it makes a connection between its two legs, connecting the pin to ground, so that we read a LOW. (The pin is still connected to 5 volts, but the resistor in-between them means that the pin is "closer" to ground.)



Figure 73 Push Button

Each port is controlled by three registers, which are also defined variables in the arduino language. The DDR register, determines whether the pin is an INPUT or OUTPUT. The PORT register controls whether the pin is HIGH or LOW, and the PIN register reads the state of INPUT pins set to input with pinMode().

**IR sensor**

The IR sensor is a 1838B IR receiver. Whenever a button on the remote is pressed, it will send an infrared signal to the IR sensor in the coded form. The IR sensor will then receive this signal and will give it to the Arduino. An infrared sensor is an electronic instrument that is used to sense certain characteristics of its surroundings. It does this by either emitting or detecting infraredradiation. Infrared sensors are also capable of measuring the heat being emitted by an object and detecting motion [20].

An [infrared sensor](https://www.elprocus.com/ir-remote-control-basics-operation-application/) is an electronic device that emits in order to sense some aspects of the surroundings. An IR sensor can measure the heat of an object as well as detects the motion. These types of sensors measure only infrared radiation, rather than emitting it that is called as a [passive IR sensor](https://www.elprocus.com/passive-infrared-pir-sensor-with-applications/). Usually in the infrared spectrum, all the objects radiate some form of thermal radiations. These types of radiations are invisible to our eyes, that can be detected by an infrared sensor. The emitter is simply an IR LED ([Light Emitting Diode](https://www.elprocus.com/explain-different-types-leds-working-applications-engineering-students/)) and the detector is simply an IR photo diode which is sensitive to IR light of the same wavelength as that emitted by the IR LED. When IR light falls on the photo diode, The resistances and these output voltages, change in proportion to the magnitude of the IR light received.



Figure 74 IR Sensor

**Resistor**

Is a component that resists the flow of direct or alternating electric circuit. Resistors can limit or divide the current, reduce the voltage, protect an electric circuit, or provide large amounts of heat or light. In our project we use 1000-ohm resistor used in the connection of push button.

****

Figure 75 Resistors

**Serial Communication**

This concept used for communicating Arduino and Unity 3D. One of the serial communication protocols supported by Arduino UNO is UART. This serial data transmission formats are available on Arduino Used for communication between the Arduino board and to another Arduino or processor to processor. All Arduino boards have at least one serial port (also known as a UART or USART), and some have several [9].

UART stands for Universal Asynchronous Reception and Transmission and is a simple communication protocol that allows the Arduino to communicate with serial devices. The UART system communicates with digital pin 0 (RX), digital pin 1 (TX), and with another computer via the USB port. This peripheral, found on all Arduino boards, allows the Arduino to directly communicate with a computer thanks to the fact that the Arduino has an onboard USB-to-Serial converter.

**Mechanical parts design and description**

**Horizontal wood:** horizontal bars on which the Agrobot moves. The length of the wood is 55cm by 59cm for the prototype as shown below.

**Wood frame**

The frame is made by wood rectangular materials.

Specification:

1) Material: wood

2) Dimension of the wood: 55cm x 59cm (near to Square cross-section wood plate)

3) Thickness of wood: 2.5 cm

4) Manufacturing process: manual work (i.e. by mounting each pieces of the wood components by using mounting components such as nail, cola etc.…)

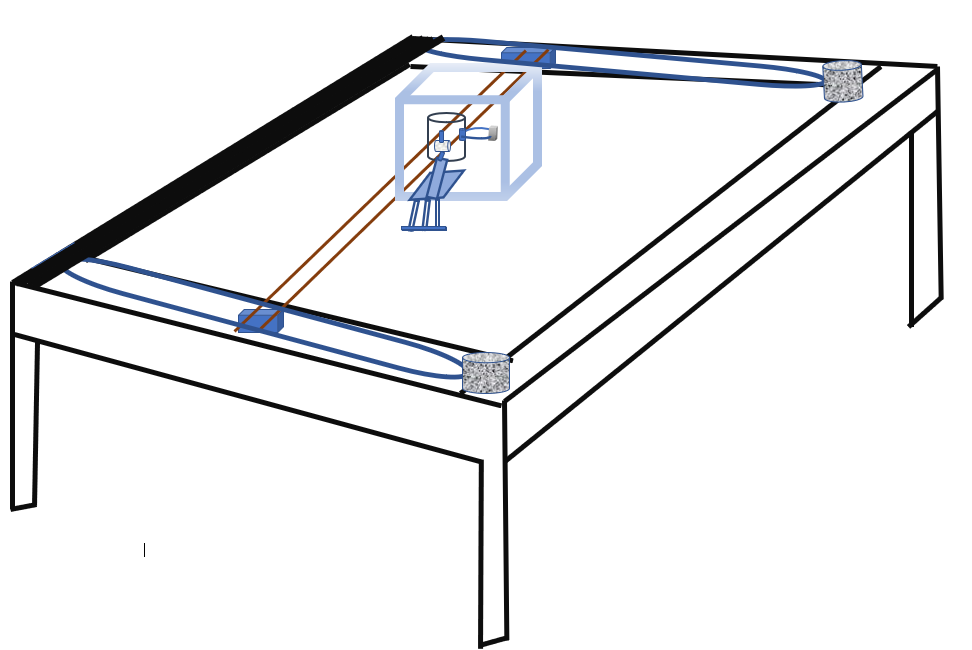
****

Figure 76 AgrobotPrototype Full Designs

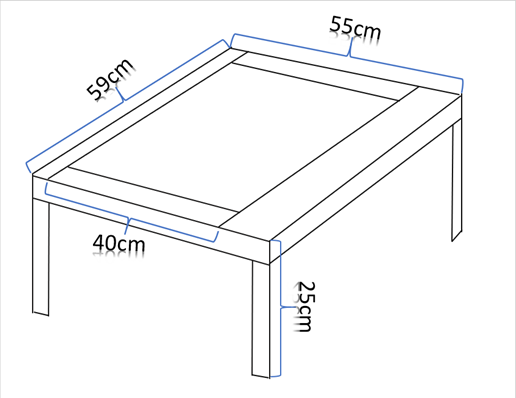
****

Figure 77 Measurements of the wood frame

**Seed dispensers**

**Seed container**

Seed container is specific container which is used for holding the seeds before it gets picked by seed dispensers. Seed container can be of any shape such as circular, rectangular, etc., but for its convenience in handling in seed sowing machine, we selected the circular shape of the seed container, designed and manufactured it.

Specification:

1. Description: 0.5 mm thick

2. Material: upper part of plastic bottle

3. Manufacturing Process: manual work

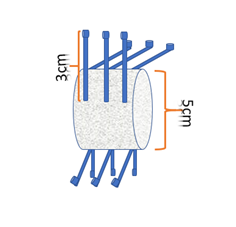
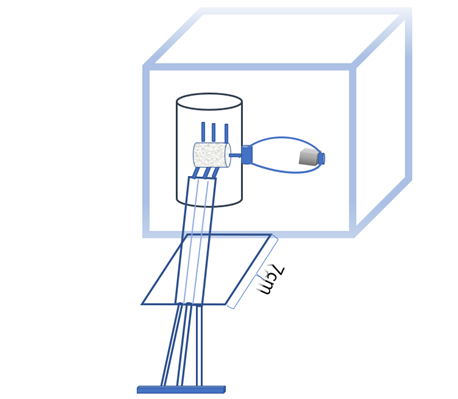


Figure 78 Seed dispenser

Figure 79 Full designs of seed container and dispenser

**Ball bearing**

A ball bearing is a type of [rolling-element bearing](https://en.wikipedia.org/wiki/Rolling-element_bearing) that uses [balls](https://en.wikipedia.org/wiki/Ball_(bearing)) to maintain the separation between the [bearing](https://en.wikipedia.org/wiki/Bearing_(mechanical)) [races](https://en.wikipedia.org/wiki/Race_(bearing)).

The purpose of a ball bearing is to reduce rotational friction and support [radial](https://en.wikipedia.org/wiki/Radius) and [axial](https://en.wikipedia.org/wiki/Axis_of_rotation) loads. It achieves this by using at least three races to contain the balls and transmit the loads through the balls. In most applications, one race is stationary and the other is attached to the rotating assembly.



Figure 80 Ball bearing

**Plastic tubes:**

Plastic tube is a tubular section, or hollow cylinder, made ofplastic and it is a circular cross-section, used mainly a path of Teff from container to the ground.



Figure 81 Plastic tube

**Wheel**

A circular frame of hard material that may be solid, partly solid, or spooked and that is capable of turning on an axle



Figure 82 Wheel

**Belt**

A belt is a loop of flexible material used to link two or more rotating shafts mechanically, most often parallel. Belts may be used as a source of motion, to transmit power efficiently or to track relative movement. Belts are looped over pulleys and may have a twist between the pulleys, and the shafts need not be parallel.



Figure 83 Belt

**Software tools**

To develop our system we have used different software tools. These are client and server side tools to develop. For the client side we have used Android, to request service of the system and the server side to give the response we have used PHP. For the embedded system parts C Language is used, for backend database we have used MYSQL.

## Prototype Setup

**Setup installed at the client side:** Android application, hardware tools (keypad, pushbutton, LCD)

The Client side programming has mostly to do with the user interface, with which the user interacts. In our project hence it is in android development it will be installed on any portable android phones. Its code is mainly done using java. The hardware’s that the client uses like keypads are for inputting, push buttons for controlling the agrobot and lcd for displaying information’s,

**Components:**

* Android
* Hardware tools

**Setup installed server side:**

Server side programming has to do with generating dynamic content. It runs on servers. The application will search a database in order to show the real time updated information on the farmlands and reports. So the application interacts with backend database using sql query (MYSQL).

**Components:**

* Apache
* Microcontroller
* MYSQL

## Implementation Detail

Table 29 Implementation detail

|  |  |
| --- | --- |
| Component name | Implementation detail |
| Software | Implemented using android and is responsible for viewing details, managing and generating report. |
| Electrical | Implemented using C, android IDE and different hardware tools for controlling speed, movement, communication between the components, manage input distance and row spacing, determine (position, level of seed, critical level, farmland coverage). |
| Mechanical | Designed and Implemented using different hardware tools (horizontal bars, seed containers) |

## Electrical Design and Implementation

The working principles (block diagrams) of our systems are as follows:

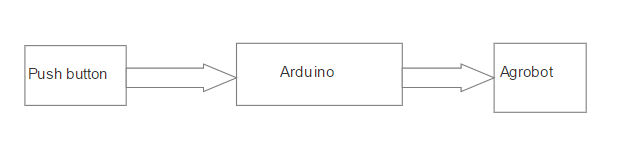


Figure 84 Block diagram for agrobot control movement

**Description for agrobot control movement**

In order to control the movement from the agrobot, whenever the corresponding push buttons are pressed based on the input and sensing values of the ultrasonic sensor the motor will move accordingly.

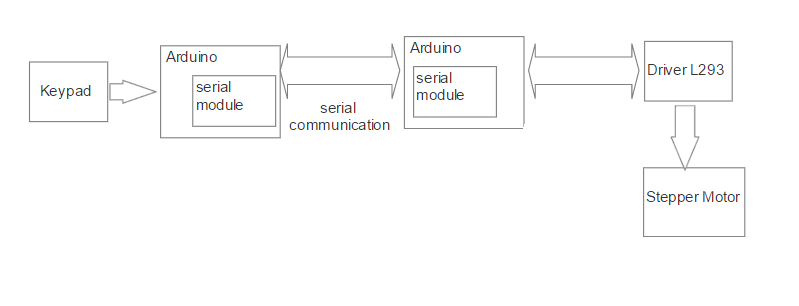


Figure 85 Block diagram for agrobot manage input

**Description of agrobot manage input**

The intended input distance will be entered to keypad and based on input and sensing values of the ultrasonic sensor the stepper rotates and the robot moves to.

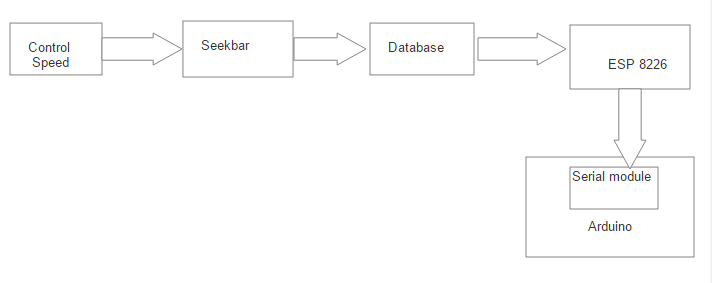


Figure 86 Block diagram app control speed

Description of application control speed

In order to control speed from the app, user uses seek bar and inserts the speed value to the database from which the ESP 8226 fetches the value and passes serially to Arduino to control the speed at which the stepper motor is rotating.

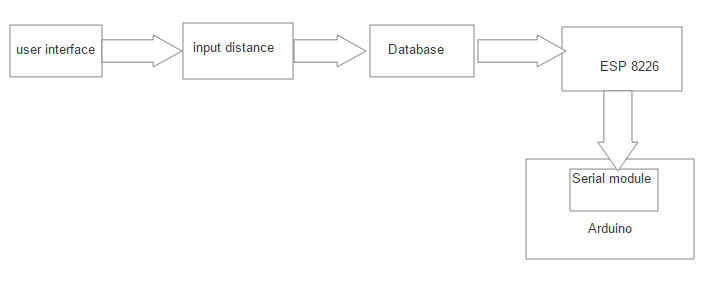


Figure 87 Block diagram app manage input distance

**Description app manage input distance**

User will insert input distance from the app to the database from which the ESP 8226 fetches and serially pass to arduino and based on the input and sensing values of ultrasonic sensor , the stepper motor rotates and robot moves to intended distance and cover.



Figure 88 Block diagram app control movement

**Description app control movement**

Whenever the movement button from the agrobot is pressed, it will be saved to database from which it is fetched by ESP 8226 and serially sent to the Arduino and based on the input and sensing values of given ultrasonic sensor servomotor rotates accordingly.

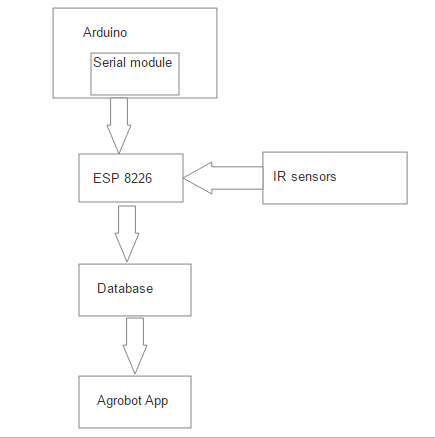


Figure 89 Block diagram critical seed notification

**Description of critical seed level block diagram**

Whenever the IR sensor Rx is not receiving any thing that means the Tx=1 or nothing is reflecting the infrared radiation the Arduino serially sends data to ESP which in turn saves to database and notifies the application.



Figure 90 Block diagram determine seed level

**Description determine seed level**

Whenever the IR sensor Rx is receiving signal that means there is teff in the container which is reflecting the infrared radiation and Arduino serially sends the sensed data to ESP which in turn saves to database.

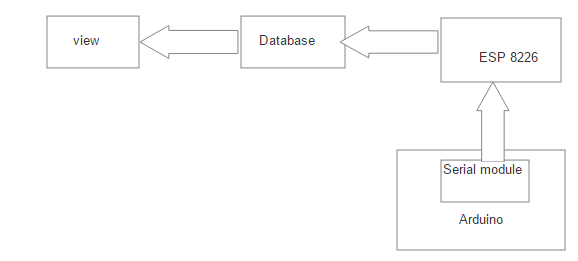


Figure 91 Block diagram of view details

**Description of view details**

Whenever view is pressed from the app, data’s sensed from the IR sensor (seed level, critical seed level) and the Ultrasonic Sensor (farmland coverage) that are stored in the database will be fetched and displayed.

# CHAPTER SIX

# SYSTEM TESTING

## Introduction

System Testing is a level of software testing where complete and integrated software is tested. The purpose of this test is to evaluate the system’s compliance with the specified requirements.

## Scope

This project mainly focuses on sowing seed on the prepared land. This system is intended to allow users to sow seed in row manner, view how much farm land is covered, user can check how much seed is in seed container, user can communicate with the robot with the help of Android application, user can control the movement of the robot on the agrobot itself.

## Resources

Resources enable us to manage resources used by our tests. So, we used the System Requirement specification document of as a resource TSA.

## Schedule

Scheduling is the time arrangements given to the responsible body of the system to perform the testing activities designed.

Table 30 schedule for testing

|  |  |  |
| --- | --- | --- |
| Type of testing | Date of testing | Tested by |
| Unit testing | July 10, 2019 | Mengistu and Abdin |
| Integration testing | July 15, 2019 | Merga and Tizita |
| System testing | July 20,2019 | Getu |

## Features to be tested and not to be tested

### Features to be tested

Are the features of the system or sub-system that will be tested. Identifies the business operations, scenarios, and functionality that are to be tested in each system and sub-system as these are what deliver value to the organization.

Table 31 Features to be tested

|  |  |  |  |
| --- | --- | --- | --- |
| no | Test targets | Feature to be tested | Tested by |
| 1 | Functional requirement | General functionality of the system | Merga, Tizita |
| 2 | Code | Each code component of the software | Mengistu |
| 3 | Hardware status | Testing all hardware part functionality | Getu , Abdin |

### Features not to be tested

Define which of the features of the system or sub-system that will NOT be tested. This is necessary so as to avoid later confusion when stakeholders thought something would be tested, but was not. These features includes:-

* Non Functional requirements Except Security and usability.
* Exact response time of the system

## Pass/fail criteria

It specifies the criteria that will be used to determine whether each test item (software/product) has passed or failed testing.

Table 32 Pass fail criteria

|  |  |  |  |
| --- | --- | --- | --- |
| Functional requirement | Expected result | Actual outcome | Pass/fail |
| Login | Pass | Pass | Pass |
| Manage input distance | Pass | Pass | Pass |
| Control movement | Pass | Pass | Pass |
| Report | Pass | Pass | Pass |
| View detail | Pass | Pass | Pass |
| Critical seed level | Pass | Pass | Pass |

**Approach**

A test approach is the test strategy implementation of a project, defines how testing would be carried out. And in our project we have used proactive approach the test design process is started as early as possible in order to find and fix the defects before the build is created.

## Test case specification

Table below shows the test specifications of functional requirements used to write the test cases along with the test case numbers for each test case and a short description of the test cases.

Table 33 Test case specification

|  |  |  |
| --- | --- | --- |
| TestCase Id | TestCase Name | Description |
| TSA\_TC 1 | Login | The User will login with a correct Username and Password; otherwise System will prompt an error message. |
| TSA\_TC 2 | Manage Input | To test that when user inputs distance to sow the robot sows accordingly. |
| TSA\_TC 3 | Control movement | To test that when user press control movement buttons the agrobot moves accordingly. |
| TSA\_TC 4 | Report | To test that the app generates the report. |
| TSA\_TC 5 | View detail | To test the system displays the required details accordingly. |
| TSA\_TC 6 | Critical seed level | To check that the system will notify when critical seed level is reached. |

The following list includes the steps that should be taken by the user, the conditions that should be met for the successful execution of the test case, and the end result that should be met for the test cases to pass.

* + - 1. TSA\_TC 1: Login
* Input: Username and Password.
* Output: valid destination Activity.
* Valid range: Username=Alphanumeric, password=capital, small letter and number.
* End message result:

1. If (User = = valid), a success message is displayed and redirected to Home activity.
2. If (User! = valid), system prompts error message on the login activity.
   * + 1. TSA\_TC 2: manage input distance

* Precondition: Agrobot must be active and working.
* Input: the distance to be covered.
* Valid range: from 0 to 300
* Expected output: The agrobot will cover the inputted distance and sow if the entered input is valid.
* Produced output:

1. If (distance == valid), then the agrobot covers the inputted distance and sow.
2. If (distance! = valid), then the system prompts an error message.
   * + 1. TSA\_TC 3: control movement

* Precondition: Agrobot must be active and working.
* Input: Forward, Backward.
* Expected output: The agrobot will move forward and backward based on the pressed button.
* Produced output:

1. If (forward == pressed), then the agrobot moves to forward direction.
2. If (backward! = pressed), then the agrobot moves to backward direction.
3. TSA\_TC 4: Report

* Precondition: Details must be in the database.
* Input: user clicks report.
* Expected output: displays the lists of data from the database.
* Produced output: report on the data’s from the database is displayed on the system.

1. TSA\_TC 5: View detail

* Precondition: Agrobot must be active and working.
* Input: user clicks on view details.
* Expected output: the system will display the seed level, position from the database.
* Produced output: the seed level, position of the robot is displayed on the system.

1. TSA\_TC 6: critical seed level

* Precondition: Agrobot must be active and working.
* Expected output: notify the system when critically low seed level is left in the container.
* Produced output: when critical seed level is reached notifies the application.

## Estimated risk and contingency plan

Table 34 Risk and contingency plan

|  |  |
| --- | --- |
| Estimated Risks | Contingency Plan |
| Lack of knowledge on designing | Allocate more time on designing working without being bored. |
| Communicating each of the hardware | Allocate more time on implementing the communication of hardware’s. |
| Virus attacks | Having copies of the projects on different materials (computers, hard disks, flash’s) |
| Lack of different hardware tools | Buying/ borrowing from outside campus. |

# CHAPTER SEVEN

# Conclusion and Recommendation

## Conclusion

As described above the system we are going to develop is Teff Sowing Agrobot which sows row wise. The benefit of our system is that the robot is small and electrically operated so that there is no need of mechanical support from human or animal. It also provides row spacing, speed controlling, and coverage calculation mechanism and mainly it increases productivity, minimize the work of farmer and also reduces the time for seed sowing operation.

While developing TSA we have gained more knowledge on how to design and assemble hardware modules, programming languages like c (for the embedded system parts), Android. Overall, we can say that we have implemented what we gain from theoretical part of knowledge to practical work.

## Recommendation

For the further improvements of the system the following points can be recommended:

* The battery of the agrobot can be replaced with solar system.
* The number of rows sowed at a time can be increased.
* Using drone for sowing.
* Automating the placement of robot from one row to the succeeding row
* Giving training to the users of the system in a country wide.
* Generally, we recommend improving the system we built in efficient and effective way.

# Annex

**Sample code for login controller**

package com.example.mrga.agrobotapp;

import android.content.Intent;

import android.support.v7.app.AppCompatActivity;

import android.os.Bundle;

import android.view.View;

import android.widget.Button;

import android.widget.EditText;

import android.widget.TextView;

import android.widget.Toast;

import com.android.volley.AuthFailureError;

import com.android.volley.Request;

import com.android.volley.RequestQueue;

import com.android.volley.Response;

import com.android.volley.VolleyError;

import com.android.volley.toolbox.StringRequest;

import com.android.volley.toolbox.Volley;

import com.basgeekball.awesomevalidation.AwesomeValidation;

import org.json.JSONException;

import org.json.JSONObject;

import java.util.HashMap;

import java.util.Map;

public class SignInActivity extends AppCompatActivity {

SessionManager sessionManager;

Button login;

EditText username,password;

TextView text;

private RequestQueue requestQueue;

private static final String URL = "http://10.240.72.23:80/agrobot/user\_control.php";

private StringRequest request;

AwesomeValidation awesomevalidation;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_sign\_in);

sessionManager = new SessionManager(getApplicationContext());

if(sessionManager.loggedin()){

startActivity(new Intent(getApplicationContext(),MainActivity.class));

finish();

}

valid();

text=(TextView)findViewById(R.id.signup);

login=(Button)findViewById(R.id.login);

requestQueue = Volley.newRequestQueue(this);

text.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

startActivity(new Intent(SignInActivity.this,RegistrationActivity.class));

}

});

login.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

//startActivity(new Intent(SignInActivity.this,MainActivity.class));

request = new StringRequest(Request.Method.POST, URL, new Response.Listener<String>() {

@Override

public void onResponse(String response) {

try {

JSONObject jsonObject = new JSONObject(response);

if(jsonObject.names().get(0).equals("success")){

/\* HashMap<String,String> hashMap = new HashMap<String, String>();

hashMap.put("username",username.getText().toString());

hashMap.put("password",password.getText().toString());

// sessionManager.createSession(hashMap.get(username),hashMap.get(password));

sessionManager.createLoginSession(hashMap.get(username),hashMap.get(password));

\*/

sessionManager.setLoggedin(true);

Toast.makeText(getApplicationContext(),"SUCCESS "+jsonObject.getString("success"),Toast.LENGTH\_SHORT).show();

startActivity(new Intent(getApplicationContext(),MainActivity.class));

finish();

}else {

Toast.makeText(getApplicationContext(), "Error" +jsonObject.getString("error"), Toast.LENGTH\_SHORT).show();

}

} catch (JSONException e) {

e.printStackTrace();

}

}

}, new Response.ErrorListener() {

@Override

public void onErrorResponse(VolleyError error) {

}

}){

@Override

protected Map<String, String> getParams() throws AuthFailureError {

HashMap<String,String> hashMap = new HashMap<String, String>();

hashMap.put("username",username.getText().toString());

hashMap.put("password",password.getText().toString());

//sessionManager.createSession(hashMap.get(username),hashMap.get(password));

// sessionManager.createSession(hashMap.get(username),hashMap.get(password));

/// sessionManager.createLoginSession(hashMap.get(username),hashMap.get(password));

sessionManager.createLoginSession(hashMap.get(username),hashMap.get(password));

return hashMap;

}

};

requestQueue.add(request);

}

});

}

private void valid() {

username=(EditText) findViewById(R.id.username);

password=(EditText) findViewById(R.id.pass);

if(username.getText().toString().trim().length()==0){

username.setError(getString(R.string.usernameerr));

username.requestFocus();

}

else if(password.getText().toString().trim().length()==0){

password.setError(getString(R.string.passworderr));

password.requestFocus();

}

}

}

**For Report**

@Override

public View onCreateView(LayoutInflater inflater, ViewGroup container,

Bundle savedInstanceState) {

View reportt= inflater.inflate(R.layout.fragment\_report, container, false);

listView = (ListView)reportt.findViewById(R.id.listview1);

//initList();

accessWebService();

return reportt;

}

private class JsonReadTask extends AsyncTask<String, Void, String> {

@Override

protected String doInBackground(String... params) {

HttpClient httpclient = new DefaultHttpClient();

HttpPost httppost = new HttpPost(params[0]);

try {

HttpResponse response = httpclient.execute(httppost);

jsonResult = inputStreamToString(

response.getEntity().getContent()).toString();

}

catch (ClientProtocolException e) {

e.printStackTrace();

} catch (IOException e) {

e.printStackTrace();

}

return null;

}

private StringBuilder inputStreamToString(InputStream is) {

String rLine = "";

StringBuilder answer = new StringBuilder();

BufferedReader rd = new BufferedReader(new InputStreamReader(is));

try {

while ((rLine = rd.readLine()) != null) {

answer.append(rLine);

}

}

catch (IOException e) {

// e.printStackTrace();

Toast.makeText(getActivity(),

"Error..." + e.toString(), Toast.LENGTH\_LONG).show();

}

return answer;

}

@Override

protected void onPostExecute(String result) {

ListDrwaer();

}

}

public void accessWebService() {

JsonReadTask task = new JsonReadTask();

// passes values for the urls string array

task.execute(new String[] { url });

}

public void ListDrwaer() {

List<Map<String, String>> reportList = new ArrayList<Map<String, String>>();

try {

JSONObject jsonResponse = new JSONObject(jsonResult);

JSONArray jsonMainNode = jsonResponse.optJSONArray("emp\_info");

for (int i = 0; i < jsonMainNode.length(); i++) {

JSONObject jsonChildNode = jsonMainNode.getJSONObject(i);

String name = jsonChildNode.optString("sowDate");

String number = jsonChildNode.optString("Area");

String outPut = name + "-" + number;

reportList.add(createReport("report", outPut));

}

} catch (JSONException e) {

Toast.makeText(getActivity(), "Error" + e.toString(),

Toast.LENGTH\_SHORT).show();

}

SimpleAdapter simpleAdapter = new SimpleAdapter(this.getActivity(), reportList,

android.R.layout.simple\_list\_item\_1,

new String[] { "report" }, new int[] { android.R.id.text1 });

listView.setAdapter(simpleAdapter);

}

private HashMap<String, String> createReport(String name, String number) {

HashMap<String, String> employeeNameNo = new HashMap<String, String>();

employeeNameNo.put(name, number);

return employeeNameNo;

}

}

For control

package com.example.mrga.agrobotapp.fragments;

import android.os.Bundle;

import android.support.v4.app.Fragment;

import android.view.LayoutInflater;

import android.view.View;

import android.view.ViewGroup;

import android.widget.Button;

import android.widget.Toast;

import com.android.volley.AuthFailureError;

import com.android.volley.Request;

import com.android.volley.RequestQueue;

import com.android.volley.Response;

import com.android.volley.VolleyError;

import com.android.volley.toolbox.StringRequest;

import com.android.volley.toolbox.Volley;

import com.basgeekball.awesomevalidation.AwesomeValidation;

import com.example.mrga.agrobotapp.R;

import org.json.JSONException;

import org.json.JSONObject;

import java.util.HashMap;

import java.util.Map;

/\*\*

\* A simple {@link Fragment} subclass.

\*/

public class ControlFragment extends Fragment {

AwesomeValidation awesomevalidation;

Button btn1,btn2;

private static String URL\_CNTRL= "http://192.168.137.13:80/agrobot/control.php";

@Override

public View onCreateView(LayoutInflater inflater, ViewGroup container,

Bundle savedInstanceState) {

// Inflate the layout for this fragment

View contrl= inflater.inflate(R.layout.fragment\_control, container, false);

btn1 =(Button)contrl.findViewById(R.id.forward);

btn2 =(Button)contrl.findViewById(R.id.backward);

btn1.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

forward();

}

});

btn2.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

backward();

}

});

return contrl;

}

public void forward(){

final String forw = "1";

StringRequest stringRequest = new StringRequest(Request.Method.POST, URL\_CNTRL,

new Response.Listener<String>() {

@Override

public void onResponse(String response) {

try{

JSONObject jsonObject = new JSONObject(response);

String success = jsonObject.getString("success");

if(success.equals("1")){

//Toast.makeText(ManageFragment.this,"inputted successfully",Toast.LENGTH\_SHORT).show();

Toast.makeText(getActivity(),"going forward!",Toast.LENGTH\_SHORT).show();

}

}

catch (JSONException e) {

e.printStackTrace();

}

}

},

new Response.ErrorListener() {

@Override

public void onErrorResponse(VolleyError error) {

}

})

{

@Override

protected Map<String, String> getParams() throws AuthFailureError {

Map<String,String> params = new HashMap<>();

params.put("distance",forw);

return params;

}

};

RequestQueue requestQueue = Volley.newRequestQueue(this.getActivity().getApplicationContext());

requestQueue.add(stringRequest);

}

public void backward(){

final String forw = "0";

StringRequest stringRequest = new StringRequest(Request.Method.POST, URL\_CNTRL,

new Response.Listener<String>() {

@Override

public void onResponse(String response) {

try{

JSONObject jsonObject = new JSONObject(response);

String success = jsonObject.getString("success");

if(success.equals("1")){

//Toast.makeText(ManageFragment.this,"inputt

ed successfully",Toast.LENGTH\_SHORT).show();

Toast.makeText(getActivity(),"going backward!",Toast.LENGTH\_SHORT).show();

}

}

catch (JSONException e) {

e.printStackTrace();

}

}

},

new Response.ErrorListener() {

@Override

public void onErrorResponse(VolleyError error) {

}

})

{

@Override

protected Map<String, String> getParams() throws AuthFailureError {

Map<String,String> params = new HashMap<>();

params.put("distance",forw);

return params;

}

};

RequestQueue requestQueue = Volley.newRequestQueue(this.getActivity().getApplicationContext());

requestQueue.add(stringRequest);

}

}

**Sample code for the hardware**

#include <ESP8266WiFi.h>

#include <WiFiClient.h>

#include <ESP8266WebServer.h>

#include <ESP8266HTTPClient.h>

#include <SoftwareSerial.h>

SoftwareSerial s(D6,D5);

#include <ArduinoJson.h>

/\* Set these to your desired credentials. \*/

const char \*ssid = "leta"; //ENTER YOUR WIFI SETTINGS

const char \*password = "135792468";

//Web/Server address to read/write from

const char \*host = "192.168.137.167";

//const char \*host = "192.168.137.247";

//=======================================================================

// Power on setup

//=======================================================================

void setup() {

pinMode(D0,INPUT);

pinMode(D1,INPUT);

Serial.begin(9600);

s.begin(9600);

while (!Serial) continue;

delay(1000);

// Serial.begin(115200);

WiFi.mode(WIFI\_OFF); //Prevents reconnection issue (taking too long to connect)

delay(1000);

WiFi.mode(WIFI\_STA); //This line hides the viewing of ESP as wifi hotspot

WiFi.begin(ssid, password); //Connect to your WiFi router

Serial.println("");

Serial.print("Connecting");

// Wait for connection

while (WiFi.status() != WL\_CONNECTED) {

delay(500);

Serial.print(".");

}

//If connection successful show IP address in serial monitor

Serial.println("");

Serial.print("Connected to ");

Serial.println(ssid);

Serial.print("IP address: ");

Serial.println(WiFi.localIP()); //IP address assigned to your ESP

}

//=======================================================================

// Main Program Loop

//=======================================================================

void loop() {

//insertData(rowspacing,farmland);

insertData();

fetchingData();

// recivedData();

}

void insertData(){

// put your main code here, to run repeatedly:

HTTPClient http; //Declare object of class HTTPClient

String postion, SeedLevelString,RowSpacingString,CriticalLevelString,farmlandcoverageString,postData;

postion="8,45";

int SeedLevel=digitalRead(D0);

int rowspacing=20;

int CriticalLevel=digitalRead(D1);

int farmland=800;

//int SeedLevel=20192;

SeedLevelString = String(SeedLevel);

RowSpacingString = String(rowspacing);

CriticalLevelString = String(CriticalLevel);

farmlandcoverageString=String(farmland);

//Post Data

postData = "postion=" + postion + "&SeedLevel=" + SeedLevelString + "&RowSpacing=" + RowSpacingString + "&CriticalLevel=" + CriticalLevelString + "&farmlandcoverage=" +farmlandcoverageString;

http.begin("http://192.168.137.167/Agrobot/InsertDB.php"); //Specify request destination

http.addHeader("Content-Type", "application/x-www-form-urlencoded"); //Specify content-type header

int httpCode = http.POST(postData); //Send the request

//int httpCode1 = http.POST(postData1);

String payload = http.getString(); //Get the response payload

Serial.println(postion);

Serial.println(SeedLevelString);

Serial.println(RowSpacingString);

Serial.println( CriticalLevelString);

Serial.println( farmlandcoverageString);

Serial.println(payload); //Print request response payload

http.end(); //Close connection

delay(4000); //Here there is 4 seconds delay plus 1 second delay below, so Post Data at every 5 seconds

}

void fetchingData(){

HTTPClient http; //Object of class HTTPClient

http.begin("http://192.168.137.167//Agrobot/read\_agrobot.php");

int httpCode = http.GET();

if (httpCode > 0)

{

// const size\_t bufferSize = JSON\_OBJECT\_SIZE(2) + JSON\_OBJECT\_SIZE(3) + JSON\_OBJECT\_SIZE(5) + JSON\_OBJECT\_SIZE(8) + 370;

StaticJsonBuffer<1000> jsonBuffer;

JsonObject& root = jsonBuffer.parseObject(http.getString());

int RowSpacing = root["RowSpacing"];

int InputDistance = root["InputDistance"];

int ControlValue = root["ControlValue"];

Serial.print("RowSpacing:");

Serial.println(RowSpacing);

Serial.print("InputDistance:");

Serial.println(InputDistance);

Serial.print("ControlValue:");

Serial.println(ControlValue);

}

http.end(); //Close connection

delay(6000);

}

void recivedData(){

StaticJsonBuffer<1000> jsonBuffer;

JsonObject& root = jsonBuffer.parseObject(s);

if (root == JsonObject::invalid())

return;

Serial.println("JSON received and parsed");

root.prettyPrintTo(Serial);

Serial.print("Data 1 ");

Serial.println("");

int data1=root["data1"];

Serial.print(data1);

Serial.print(" Data 2 ");

int data2=root["data2"];

Serial.print(data2);

Serial.println("");

Serial.println("---------------------xxxxx--------------------");

}

# Prototype development

**SIMULATION RESULT AND ANALYSIS**

* + - 1. Determine the seed level and the position of agrobot on Proteus

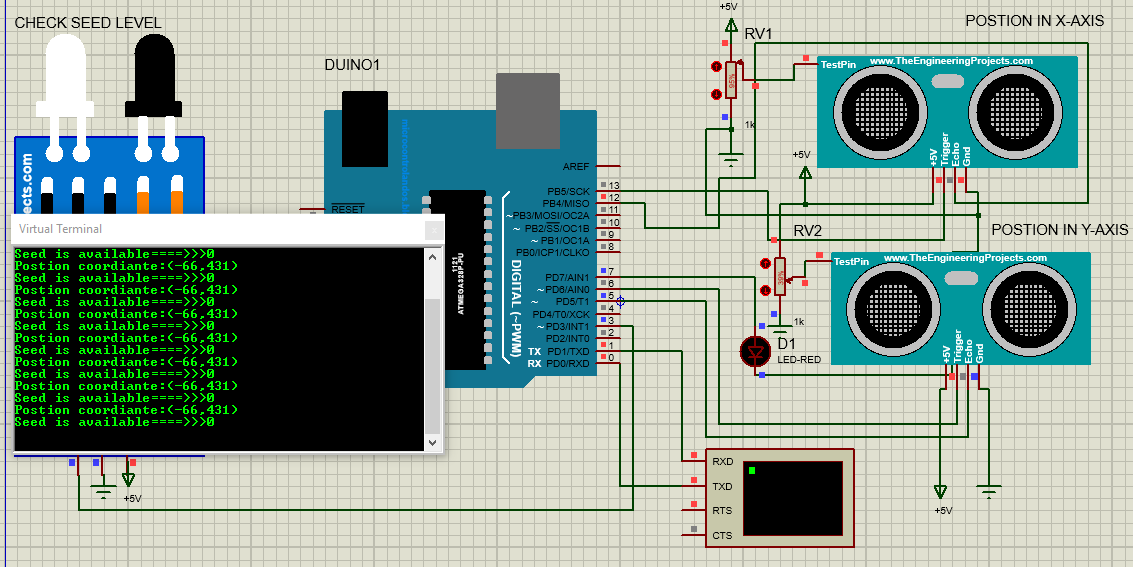
****

Figure 92 Determine the seed level and the position of agrobot on Proteus

Description:

The above sensors are used for determining position, farmland coverage (Ultrasonic sensors) in the (x, y) coordinate and the critical seed level and seed level (IR sensors).

* + - 1. user insert Input Distance to sow and display on the lcd on Proteus

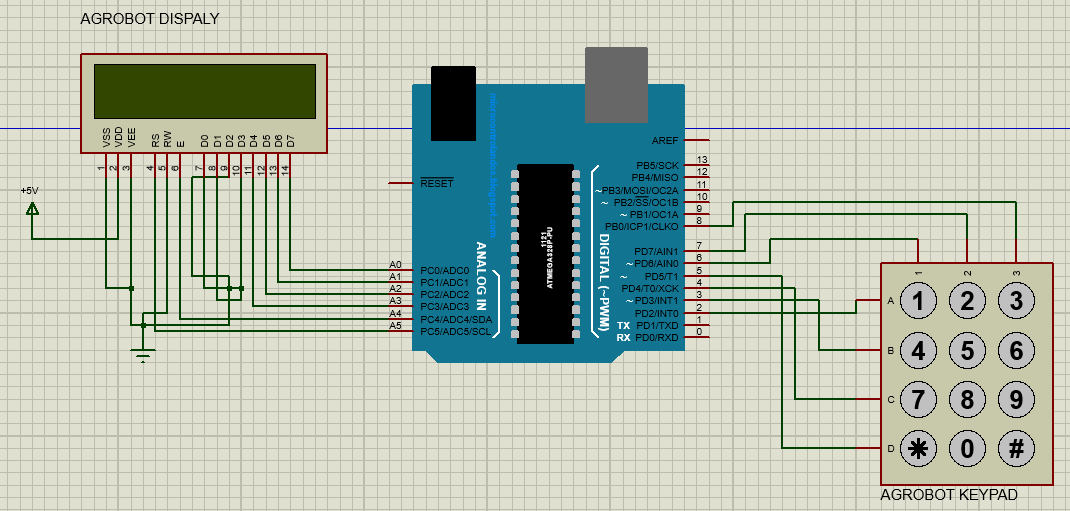
****

Figure 93 when system is off

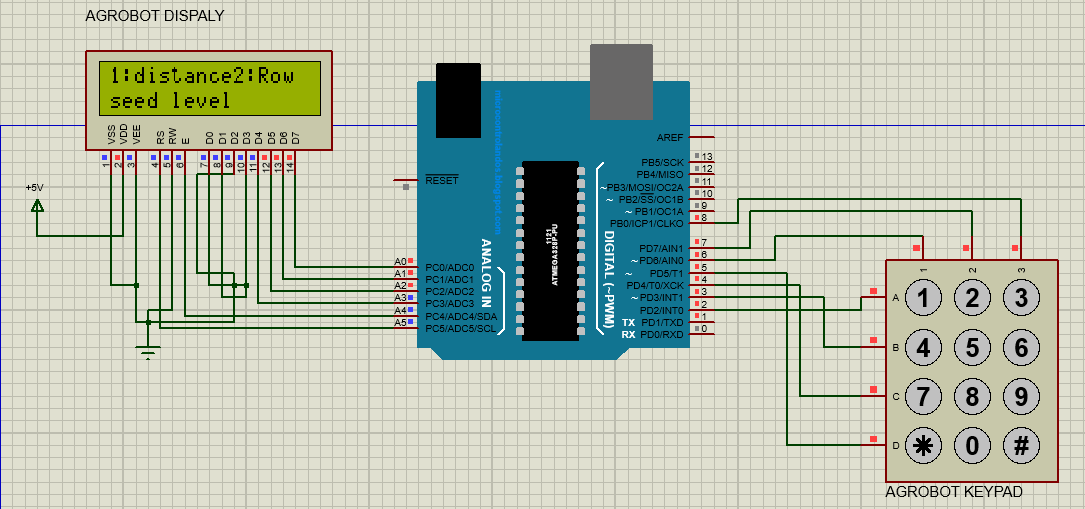
****

Figure 94 user inserts Input Distance to sow and display on the lcd on Proteus

.

Description:

The above simulation result shows user uses a keypad to insert input distance to sow the seed or to insert the row spacing user can see two options 1.for input distance 2.for row spacing user can choose one of the two options and they can insert the value based on the selected option.

* + - 1. Control the movement of the agrobot

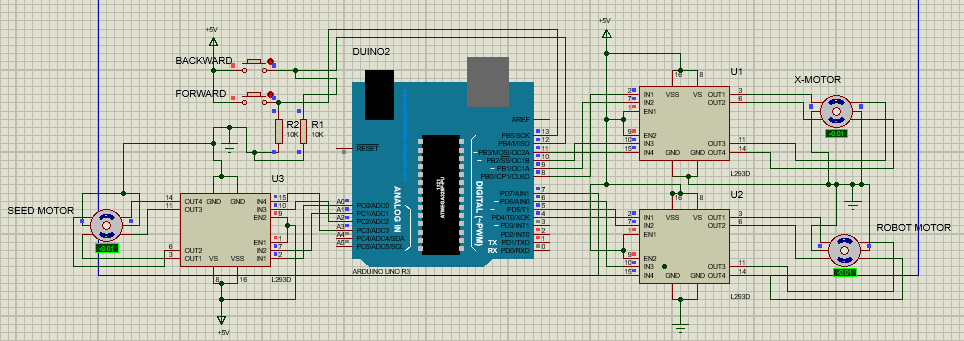


Figure 95 Control movement

Description:

The above circuit describes how the stepper motors are interfaced with Arduino to control every movement of the robot.

**Sample Interfaces of Agrobot Control App**



Figure 96 Manage input distance and row spacing Interface for English



Figure 97 Manage input distance and row spacing Interface for Amharic



Figure 98 Manage input distance and row spacing Interface for Afaan Oromoo

# References

[1] <https://www.researchgate.net/publication/325593828_Design_and_development_of_seed_sowing_AGROBOT>, Access date (march 11,2019)

[2] <https://whatis.techtarget.com/definition/agbot-agricultural-robot>, Access date(march 11,2019)

[3] <https://www.iaeme.com/MasterAdmin/uploadfolder/IJMET_09_04_103/IJMET_09_04_103.pdf>, Access date (march 11,2019)

[4] <https://www.academia.edu/31862725/Behailu_Getu>, Access date (march 11,2019)

[5] Fufa et al. 2011

[6] Assessment of Factors Affecting Farmers’ Adoption level of Row Planting Technology and Yield Improvement on the Production of Eragrostis Teff [ZUCC.]: The Case of Minjar Shenkora Woreda, Amhara Region, Ethiopia by Behailu Getu Desta.

[7]<https://www.eeaecon.org/sites/default/files/forms/Adoption%20and%20Impact%20of%20Row%20Planting.pdf>, Access date(April 01,2019)

[8] <http://www.nuruinternational.org/blog/agriculture/planting-maize-beans-wheat-and-teff-in-ethiopia/>, Access date(march 10,2019)

[9] <http://www.ifpri.org/publication/row-planting-teff-ethiopia-impact-farm-level-profitability-and-labor-allocation>, Access date (march 01, 2019)

10] https://www.arduino.cc/en/guide/introduction, Access date (July 01, 2019)

[11] https://learn.sparkfun.com/tutorials/what...arduino/whats-on-the-board, Access date (July 01, 2019)

[12] https://www.tutorialspoint.com/arduino/arduino\_stepper\_motor.htm, Access date (July 10, 2019)

[13] https://www.arduino.cc/en/tutorial/stepperSpeedControl, Access date (july10 2019)

[14] https://www.instructables.com/id/Basic-LCD-Project-Arduino-LCD-16x2-Display/, Access date (July06, 2019)

[15] https://www.electroschematics.com/12446/arduino-with-keypad/, Access date (july06, 019)

[16] http://wiring.org.co/learning/tutorials/breadboard/, Access date (July 10, 2019)

[17] https://www.tutorialspoint.com/arduino/arduino\_ultrasonic\_sensor.htm, Access date (July 02, 2019)

[18] https://en.wikipedia.org/wiki/NodeMCU, Access date (July 01, 2019)

[19] https://www.arduino.cc/en/tutorial/button, Access date (July 01, 2019)

[20] https://www.elprocus.com/infrared-ir-sensor-circuit-and-working/, Access date (July 01, 2019)