Palestine Polytechnic University
College of Information Technology and Computer Engineering

Graduation Project entitled:

Intelligent Weeding Robot Using Deep-Learning

Submitted in partial fulfillment of the requirements for the Bachelor's Degree in the College of Information Technology and Computer Engineering.

Project Team:

Alia Zatari   Bilal Hoor   Niveen Nasereddin

Supervisor:

Dr. Mousa Alrefayah.

Jun. 2022

Acknowledgment

In the name of "Allah", the most beneficent and merciful who gave us strength, knowledge and helped us to get through this project. For those who deserve our thanks the most, our parents, we are indebted to you for the rest of our lives for your unconditional love and support.

We know that thanks are not enough and there are not enough words to describe how thankful we are.

To our families and friends, thank you for your endless encouragement all our lives and especially during the completion of this project.

We would like to thank our supervisor of this project, Dr.Mousa for his valuable help and advice during this project.

We also thank our faculty and Professors at the College of Information Technology and Computer Engineering for their hard work and support to the students.

Abstract

Weeding is the most challenging process in agriculture. Traditional weeding requires a lot of labor, time and money. So, there is a need to develop an autonomous weeding method that reduces the time and cost of the weeding process without human intervention.

Our system is designed to detect weeds using a deep learning model in order to eliminate the weeds using a laser beam, which is an effective weed control method that is echo-friendly and does not harm the environment. The laser weeding precisely targets the weeds and eliminates them without affecting the crop plants.

The solution we suggest is a smart machine, based on a mobile application that processes the input images from the camera module to classify the detected objects in the image as weeds or not. After detecting the weed, the center point coordinates of the detected weed are sent to the Arduino microcontroller via the Bluetooth module, then the robot will move in 2-dimensionals to reach the position of the weed in the real environment, and activate the laser on the weed for an amount of time.

Table of Contents

List of Figures

List of Tables

List of Equations

Chapter 1: Introduction

1.1 Overview

Weeds are defined as the unwanted plants that grow among crops [1]. The process of removing such unwanted plants is called weeding. Weeds cause competition in nutrients, moisture (water), and light, and have a significant impact on the early growth and survival of crop plants [2].

In this project, we will design and build a smart agriculture machine to solve the weeding problem. This robot will have the ability to move easily on the row since it will be designed with four wheels, and will have a space between the wheels appropriate with the row width, as well as the height of the robot will be suitable with the plant's height. The robot will move in a one-way path and use a mobile application to detect if there is a weed interleaved with the plant's row, and then will eliminate the weeds precisely by the laser.

1.2 Motivation

The most important motivation is the replacement of hand weeding so this helps farmers overcome the difficulties they face in removing weeds, also using the robot will reduce herbicide dosage, because excessive spraying can damage other plants (non-target plants), increasing CO2 emission, and the herbicide residue may leach into groundwater. There is another motivation we want to achieve through the robot, which is the ability to work both day and night, and therefore reduce the dependency on farmers and decrease the costs by decreasing labor requirement.

1.3 Problem Statement

Weeds have a key impact on everyone in the world because they reduce crop productivity and quality. Intra-row weeds (the weeds that are interleaved with the crops) are difficult to eliminate in a traditional autonomous way. Also, the use of selective herbicides does not solve the problem, because chemical weeding is harmful to the environment when used excessively and increases the cost of the weeding process.

There is a need in Palestine to improve farming methods, and to introduce new techniques and machines that have a good impact on the crop yield, and the thing that sometimes we don't pay attention to, is that weeds have a great effect on the crop yield. So, we need new methods in weeding that can help farmers.

1.4 System Description

The system is a weeding robot, and it has the ability to detect the weed, in order to carry out the weeding process autonomously. This robot contains these components: a mobile application which we will deploy the deep learning model on, it will capture images of the crop row, four wheels for movement in one direction in the row without steering, a laser light attached with an arm, which moves in one dimension and a battery to power the robot. The wheels' movement is done by two Nema 17 bipolar stepper motors that are appropriate with the torque needed, and to control the position of the robot in the y-axis. The arm horizontal movement is done by a unipolar stepper motor to control the location of the laser in the x-axis.

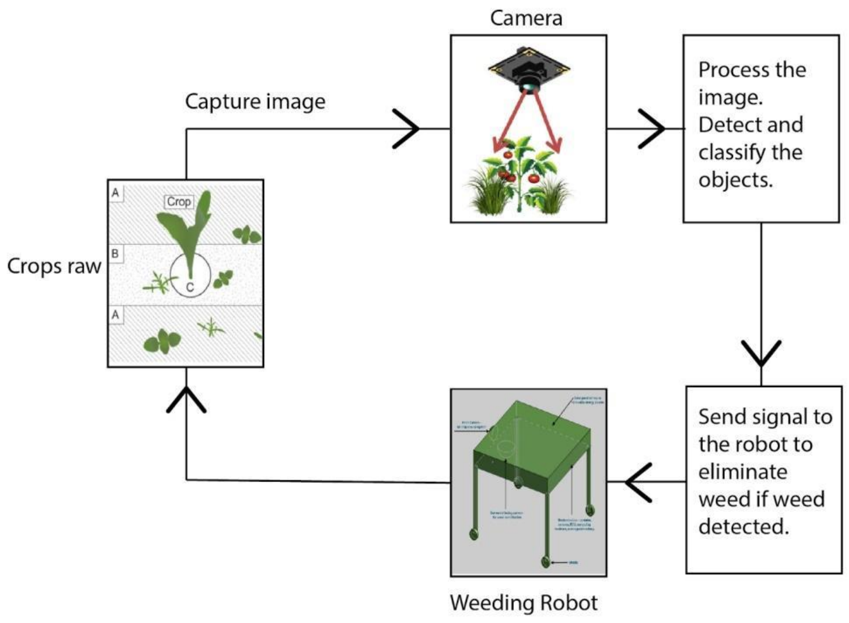

Camera
Capture image
Process the image. Detect and classify the objects.
Crop
A
B
C
A
Crops raw
Send signal to the robot to eliminate weed if weed detected.
Weeding Robot

Figure 1: Basic Block Diagram of the system

1.5 Objectives

We aim in our project to achieve several objectives such as:

1. Design a mobile robot that precisely guided through the crop rows without damaging the plants.
2. Identify intra-row weeds accurately using the concept of object detection in Deep Learning.
3. Implement a mobile application to hold the machine-learning model.
4. Design a method for estimation of the position of the identified herbs.

5. The robot will remove the weeds by directing a laser beam above it.
6. The robot must not cross the field/row boundaries.

1.6 Requirements

1.6.1 Functional Requirements:

#	Requirement	Description
1	Weeds detection	The system must be able to detect the weed. The mobile application will recognize if there are weeds or not by a deep learning model deployed on it.
2	Locating weeds	The system must be capable of specifying the location of the weeds.
3	Laser light	After identifying the weed, the system must be able to direct a laser beam at the weeds.
4	Robot movement	It should be able to move on a single-row path with the help of its four wheels.
5	Arm movement	After detecting the weeds, the arm must be able to move in one direction and stop exactly above the weed.

Table 1: System Functional Requirements

1.6.2 Non-Functional Requirements:

#	Requirement	Description
1	Accuracy	The robot performance is mostly determined by how accurately the weeds are detected and how accurately they are eliminated. So it is necessary to ensure that the system will work with high accuracy.

2	Response time	Ensuring that the system will have a low response time of weeds detection and elimination by taking the time that Phone consumes for recognizing the weeds obtained, then adding it to the time consumed by the robot to take the appropriate action.
3	Speed	The robot movement speed shall be appropriate with the time needed to capture images of the crop row and decide if there are weeds, in order for the robot to have accurate response to the true position.
4	Power Consumption	The robot weight shall be not too heavy in order to minimize the power needed to robot movement.

Table 2: System Non-Functional Requirements

1.7 Expected Results

1. We expect to develop a prototype of a weeding robot that can be developed to help farmers to reduce the effort, time and cost of the weeding process.
2. We expect that our model can identify weeds with acceptable accuracy.
3. We expect that the robot after the detection of the weed will respond in real time and direct a laser beam toward the weed.

1.8 Overview of the rest of the report

The next chapter, "Background," covers theoretical background and Literature Review, software and hardware design options, and design constraints and limitations.

The third chapter, titled "System Design," contains a full conceptual description of the system (Hardware and Software aspects), detailed design, schematic diagrams, block diagrams, structural diagrams, and any necessary information relevant to the system design.

Chapter 2: Background

2.1 Overview

In this chapter we will go over the theoretical background by demonstrating prior work and techniques that others have utilized in their projects. This chapter introduces the project's literature review, a description of the technologies employed in the project, and a description of the system's hardware and software components. Certain design specifications and constraints are described at the end of this chapter.

2.2 Theoretical background

In this section, we will provide a clear idea about the project functionality, and give some information about the general terms.

2.2.1 Robotics in Agriculture and Industrial Revolution 4.0

The fourth industrial revolution combines artificial intelligence and big data, which have gained significant attention and popularity in precision farming applications such as monitoring, diagnosing insect pests, measuring soil moisture, diagnosing harvest time, monitoring crop health status, reducing complicated human monitoring and so on [3]. Industry is extending precision agriculture with artificial intelligence and robotic technology, and its applications are embedded into smart observation that retrieves real-time information from field level data with little or no human involvement. The fourth industrial revolution is providing significant and sustainable advances to both production and agro processing through smart agricultural technologies. Also the fourth industrial revolution increases the value of farms while also extending their productivity [3].

2.2.2 Smartphones in Robots

Most robot engineers use microcontrollers or computers in order to implement the design of the robot system, and purchase off the shelf sensors like cameras to build the robot’s sensing system. Smartphones in every new release are becoming more powerful and equipped with several accessories that are useful for robots. In our project we aim to turn a smartphone into a weeding robot instead of purchasing a microcontroller with camera and other accessories that are already embedded in each new smartphone.

2.2.3 Weed Detection Methodology

Weed management in crops is a challenging task for farmers and poses a significant threat to crop yields if not done properly [4, 5]. The following alternatives to determine where intra-row weeding has to be performed were taken into consideration:

2.2.3.1 Image Processing

Most of the traditional weed detection methods based on image processing utilize the feature differences between plant leaves and weeds to distinguish them. the traditional image features and their advantages and disadvantages for the detection and recognition of four features of weeds: texture, shape, spectrum, and color.

Feature	Pros	Cons
Texture	High accuracy, strong adaptability, and robustness.	Gray-level co-occurrence matrix (GLCM takes a long time and does not meet the real-time processing requirements.
Shape	Independent of geometric translation, scaling, or rotation; robust to noise.	Shapes are deformed by disease, insect eating, and man-made or mechanical damage and incomplete under overlap and occlusion.
Color	Insensitive to the adjustment of proportion, size, and position	The color may change due to differences in soils, nutrients and sunlight. Crops and weeds with similar color will fail; leaf lesions and plant seasonality will change color.
Spectral	Robust to partial occlusion. Reflectance of crop, weeds and soil differ in the visual and near infrared wavelengths.	Plant spectral features differ at different growth stages. Spectral features are easily affected by the collection environment, and are unstable.

Table 3: A comparisons between the traditional techniques in weed classification

2.2.3.2 Deep Learning Methods and Technologies

2.2.3.2.1 Artificial intelligence and Machine Learning

Artificial intelligence (AI), manifested by machines that exhibit aspects of human intelligence (HI), is increasingly utilized in service and today is a major source of innovation. For example, robots for homes, health care, hotels, and restaurants have automated many parts of our lives; virtual bots turn customer service into self-service. AI having the ability to perform human tasks and being able to think and feel like humans, will replace human labor entirely and, thus, human interactions will fade from sight [6]. Research areas around AI applications in the workplace are related among others to machine learning and deep learning and they can be applied in industries across the globe [7].

Machine learning (ML) is a subset of the artificial intelligence domain that provides computers the ability to learn, analyze, and make their own decisions/predictions without being explicitly programmed. It is mainly categorized into predictive or supervised learning and unsupervised learning [8]. Machine Learning is frequently regarded as a difficult task because it is based on complicated data. However, Machine Learning's primary aim is to be able to predict results without having to understand every part of this complicated data. Machine Learning differs from AI in that it is mainly guided by humans. Machine learning, which includes deep learning, is a major component of artificial intelligence, as shown in figure 2.

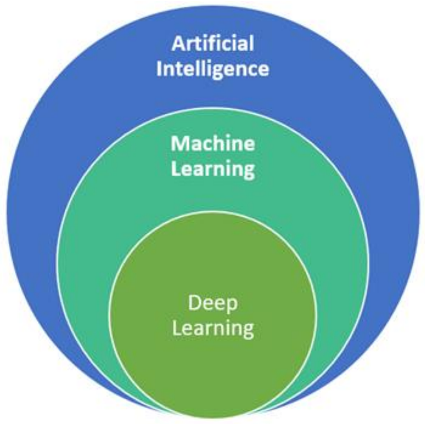


Figure 2: Artificial Intelligence, Machine Learning and Deep Learning

In Machine Learning, data is typically separated into three datasets: training set, validation set and testing set. The training set is the largest of the three, typically accounting for 70-80% of all data. It is used for learning that is to fit the parameters of the classifier. A lack of training data may lead to overfitting, meaning that a model does not generalize well to new data (i.e. the model may focus on specific patterns in the training data not relevant for newly gathered data)

[9].Then the validation set is introduced after the algorithm has learned from the training set. A validation dataset is a sample of data held back from training your model that is used to give an estimate of model skill while tuning a model's hyperparameters [10]. After the validation set, the model will go through the test set. The objective behind the test set is that the model's final performance is evaluated using data that it has never seen before. The validation dataset is different from the test dataset that is also held back from the training of the model, but is instead used to give an unbiased estimate of the skill of the final tuned model when comparing or selecting between final models [10]. After going over these three datasets, the final model should be able to predict with as much accuracy as possible. Machine Learning models might take days or weeks to train, depending on the configuration, so this method will take varying amounts of time based on the quantity of data trained in the process.

2.2.3.2.2 Artificial Neural Network and Deep Learning Algorithms

An Artificial Neural Network (ANN) is a system that is inspired by the connections of neurons in the human brain [11]. An artificial neuron is a single block mathematical entity that processes information and is essential in the functioning of a neural network [11]. Haykin stated that a typical neuron has three essential elements: a set of connection links that have their weights, a summation point, and an activation function. The neuron k can be mathematically described by the following equations [11]:

$$u_k = \sum_{j=1}^{m} w_{kj} x_j$$

$$y_k = \Phi(u_k + b_k)$$

Equation 1: Calculation of a neuron k

where u_k is linear combiner output; w_{k1}, w_{k2}, w_{k3}, . . . w_{km} are synaptic weights; x_1, x_2, x_3, . . . x_m are inputs; b_k is the bias that has the effect of lowering the input activation function; $\emptyset(.)$ is the activation function; y_k is the output of the neuron.

An artificial neural network is simply a collection of artificial neurons. Typically they are connected and organized in layers. A layer is made up of interconnected neurons that contain an activation function. A neural network consists of an input layer, an output layer, and one or more hidden layers. The input layer takes the inputs from the outside world and passes those inputs with a weighted connection to the hidden layers. The hidden layers then perform the computations and feature extractions and are activated by standard nonlinear activation functions such as tanh, ReLU (Rectified Linear Unit).

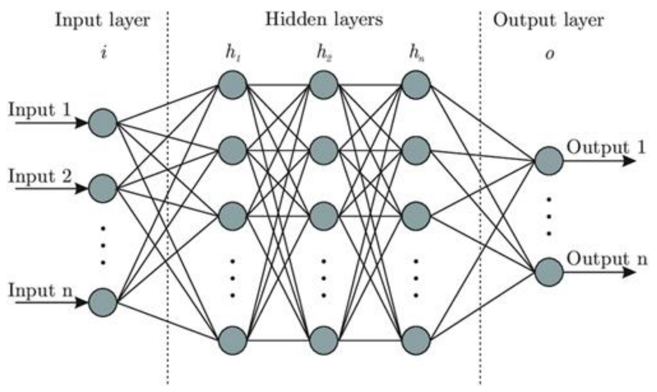


Figure 3: Artificial Neural Networks

Deep learning is a subset of machine learning that is responsible for a large portion of AI's success. Deep learning uses artificial neural networks as an underlying model for AI: while loosely based on biological neural networks such as your brain, artificial neural networks are probably best thought of as an especially nice way of specifying a flexible set of functions, built out of many basic computational blocks called **neurons**. In particular, rather than programming a specific set of instructions to solve a problem directly, deep learning models are trained on data from the real world and learn how to solve problems [12]. Deep learning is based on the concept of neural networks with several layers. The “deep” in deep learning is referring to the depth of layers in a neural network so deep learning models are often made up of tens or hundreds of these layers that have been trained with large labeled datasets and neural networks. These layers act similarly to filters in that they are made up of mathematical functions that separate the features. Each layer is made up of neurons that operate together as a unit. These neurons can be viewed as mathematical functions that take an input and combine it with the weight and activation function of the output at that layer to calculate a sum. After that, the output is moved to the subsequent layers till it reaches the output layer. Different layer types are used to build these networks. The input layer is the first layer of a neural network, and the output layer is the last. Between these layers are many hidden layers. The hidden layers are made up of nodes with weights and thresholds that have been determined. Weights represent the strength of connections between neurons, while thresholds are filters that determine whether the input signal is transferred to the next node based on activation. Starting with the input layer, the nodes in the next layer are activated based on their properties and send data to the next layer [13] [14].

2.2.3.2.3 Convolutional Neural Network (CNN)

The term convolutional neural network (CNN) refers to one of the deep neural network algorithms used to solve computer vision problems. They are often used in applications like image classification, object detection, and instance segmentation problems [15]. The features are extracted using a CNN, and then they are classified. There are four types of layers for a convolutional neural 9 network: the convolutional layer, the pooling layer, the ReLU correction layer and the fully-connected layer. CNNs are increasingly used in weed detection, and methods based on deep CNNs have achieved good results in weed detection and classification [16].

2.2.3.2.4 Transfer Learning Technique

Transfer learning is a technique that is used in many machine learning and deep learning tasks. It is a popular approach where pre-trained models are reused in a new machine learning model. If the two models are developed to perform similar tasks, then generalized knowledge can be shared between them. It has also been defined as an optimization that allows rapid progress when the model is learning for another task [17]. The transfer learning technique is proposed as the method to be utilized for the tasks of weed identification and classification as it has been 10 reported to be suitable for tasks of autonomous identification and classification tasks [18]. Negative transfer occurs when the model source domain data is dissimilar from target domain data. In other words, negative transfer can occur when the two tasks are too dissimilar [19]. As a result, the model does not perform well, leading to poor results. On the other hand, while doing transfer learning, the models are prone to overfitting, in absence of careful evaluation and tuning. Overfitting is however a general limitation for all prediction technologies [20].

2.2.3.2.5 Object Recognition

Object recognition is a general term to describe a collection of related computer vision tasks that involve identifying objects in digital photographs and it usually takes place via a neural network. These tasks can be divided into image classification, object localization, and object detection. Image classification involves assigning a class label to an image, whereas object localization involves drawing a bounding box around one or more objects in an image. Object detection is more challenging and combines these two tasks and draws a bounding box around each object of interest in the image and assigns them a class label [21].

2.2.3.2.5.1 Object Detection

Object detection is a computer technology related to computer vision and image processing, which is issued to detect instances of a class of semantic objects (such as people, buildings, or cars) in digital images and videos. It is an important task in many popular fields such as medical diagnosis, robot navigation, automatic driving, augmented reality and so on. In these complex scenarios, object detection methods based on deep learning approaches, such as R-CNN, Fast R-CNN, Faster R-CNN, YOLO and SSD show greater advantages than traditional methods [22].

When analyzing videos consisting of multiple images (frames) per second, and the detection is executed in real-time, the object detection algorithm is considered a **real-time object detection algorithm**. Analyzing multiple images per second of objects puts heavy emphasis on efficient algorithms, as the computational power required is increased [23].

Object detection models can be divided according to how many stages they need for the detection. Multi-stage detectors usually need two stages for the detection as single-stage detectors need only one. The advantage of multi-stage detectors is the accuracy they provide. However, the multi-stage detectors are too slow for real-time object detection. Single-stage detectors are often several times faster than multi-stage detectors but have had relatively low object detection accuracy. The introduction and development of single-stage object detection algorithms such as **You Only Look Once (YOLO)** and **Single Shot Detector (SSD)** have made real time object detection possible [24]. The primary difference between the two architectures is that the YOLO architecture utilizes two fully connected layers, whereas the SSD network uses convolutional layers of varying size [23].

2.2.3.2.5.1.1 You Only Look Once (YOLO)

YOLO is one of the fastest object detection methods with good real-time performance and high accuracy. The name You Only Look Once is based on how the algorithm only looks once at an image while many other algorithms need two looks. The structure of YOLO is straightforward. It can directly output the position and category of the bounding box through the neural network. The speed of YOLO is fast because YOLO only needs to put the picture into the network to get the final detection result, so YOLO can also realize the time detection of video [25].

By running on the powerful GPU platform, YOLO and its improvements have reached high accuracy and fast speed. However, the model sizes of these object detection algorithms are too large for constrained environments with limited storage memory devices and they cannot work in constrained environments with real-time performance. Tiny-YOLO algorithms are one of the improvements of YOLO with a relatively small model size for constrained environments. However, its detection accuracy is not high and the real-time performance is still not satisfactory on low computing power devices [22].

YOLO, a single-stage model, uses a convolutional neural network (CNN) to make its predictions and proposals. In the CNN, the input image is divided by YOLO into a grid with GxG cells, and the grid then generates N predictions for bounding boxes (GxGxN boxes in total). Once the image is divided, each grid undergoes classification and localization of the object. The objectness or the confidence score of each grid is found. If there is no proper object found in the grid, then the objectness and bounding box value of the grid will be zero or if there is found an object in the grid then the objectness will be 1 and the bounding box value will be its corresponding bounding values of the found object [26]. Each bounding box is limited to having only one class during the time of prediction, which restricts the network from finding smaller objects [23].

2.2.3.2.5.1.2 Single Shot Detector (SSD)

Single-shot detector (SSD) is a neural network model designed for real-time object detection. The SSD approach is based on a feed-forward convolutional network that produces a fixed-size collection of bounding boxes and scores for the presence of object class instances in those boxes, followed by a non-maximum suppression step to produce the final detections [27]. Generally SSD uses various feature layers as classifiers, in which a set of different aspect ratios in the form of default boxes at each place a convolutional way is used to evaluate each feature map. Also every classifier predicts the class scores and shape offset score with respect to the boxes [28]. SSD’s have dual mechanisms: a spinal model and SSD head. The spinal model is generally a network of pre-trained image classifications as fact-makers. This is usually a network such as MobileNet that trains over a million images that have been completely removed from the associated ranking layer. SSD Head This waist contains only one or more fixed layers, and the results are defined as classes of boxes and objects bound to the dimensional place of the closing layer activation [29].

In our project we used the YOLO algorithm, since we need a fast algorithm that works quickly in real time, also YOLO achieves more than twice the mean average precision of other real-time systems.

2.2.4 Positioning of Weeding

The following alternatives to position the weeding actuator at the location indicated by the weeding robot were taken into consideration:

- GPS: The position of the actuator can be measured by mounting a GPS antenna above the actuator position. The maximum position update frequency is not sufficient for precise actuator positioning.
- Dead reckoning: With a wheel encoder the position of the actuator can be measured [30]. Accumulation of inaccuracies over the distance between sensors and actuators occurs but is limited if the distance between both is small.
- Machine vision: A machine vision system could track both the actuator position and the position at which it should become active. To do this a specially developed image processing algorithm is needed.
- Number of steps: we can rotate the stepper motor for a certain amount of steps to reach the desired position. This is an open-loop circuit that can give us the ability to control the position with high accuracy.

In our case, we will use machine vision to determine the position and stepper motors to reach the intended position.

2.2.5 Weeding Methods

There are a variety of ways to weed control, and the following options have been considered for intra-row weeding:

2.2.5.1 Chemical Weed Control:

Chemical weeding (herbicide) is widely used as a traditional solution. However, herbicides are viewed critically because of its negative environmental effects, such as the destruction of non-target plants and beneficial insects for the soil, as well as the health issues it causes in animals and humans, such as cancer and birth defects. Weed control based on chemical treatment has increased costs and made some weeds resistant to herbicides (251 species by the International Survey of Herbicide Resistant Weeds, 2017) [31].

2.2.5.2 Non-Chemical Weed Control:

Non-chemical technologies can control weeds without harming the environment, both in organic and conventional farming, many non-chemical weeding methods have been developed ranging from electrocuting through to using mechanical or thermal methods. Mechanical and thermal machines for weed control are physical means. Mechanical methods like finger weeders, torsion weeders, and intelligent weeders are the most commonly used form of weed control in such systems, but it requires large investments in energy, labor, and time [32]. Thermal weed control with flaming is limited to selective applications. Laser technology could be an alternative tool. Lasers have been considered as a valid cutting device for physical weed control [33]. In recent years, extensive study on the effects of laser treatment as a weed control approach has been conducted. The results show that the laser may significantly reduce weed growth.

In our project, we will use laser technology to eliminate weeds, with the aim of eliminating the need of herbicides while increasing productivity. This method will eliminate the health and environmental dangers associated with herbicide use.

2.2.6 Programming Languages

We are planning to use Python as our primary programming language, since it deals with the complex algorithms in machine learning efficiently and it is also widely used by the machine learning community, in addition most of the research work we saw uses python as a primary tool to build the model that was responsible for the gestures detection.

2.2.7 Frameworks and Libraries

2.2.7.1 Keras

Keras is a minimalist Python library for deep learning that can run on top of Theano or TensorFlow. It was developed to make developing deep learning models as fast and easy as possible for research and development. It runs on Python 2.7 or 3.6 and can seamlessly execute on GPUS and CPUS given the underlying frameworks. It is released under the permissive MIT license. Keras was developed and maintained by François Chollet, a Google engineer using four guiding principles: modularity, minimalism, extensibility and Python [34].

2.2.7.2 TensorFlow and TensorFlow Lite

TensorFlow is an open source library for fast numerical computing. It was created and is maintained by Google and released under the Apache 2.0 open source license. The API is nominally for the Python programming language, although there is access to the underlying C++ API. Unlike other numerical libraries intended for use in Deep Learning like Theano, TensorFlow was designed for use both in research and development and in production systems, not least RankBrain in Google search and the fun DeepDream project. It can run on single CPU systems, GPUs as well as mobile devices and large scale distributed systems of hundreds of machines [34].

TensorFlow Lite has two components: an interpreter and a converter. On lower-powered devices, the interpreter runs optimized models. The converter converts TensorFlow models into an interpretable format. The converter also increases optimizations and efficiency.

Training models are currently unavailable in TensorFlow Lite. The model must first be trained on a computer with more processing power than the relatively low-performance end device, and then converted to a TensorFlow Lite-file. Alternatively, Google Colab, which provides an external online-based GPU with Cuda architecture, can be used to train TensorFlow models. After conversion, the trained TensorFlow Lite-file is provided to the device's interpreter [35].

2.2.7.3 Pytorch

PyTorch is an optimized tensor library for deep learning using GPUs and CPUs [36]. PyTorch can be seen as a Python front end to the Torch engine which at its heart provides the ability to define mathematical functions and compute their gradients. PyTorch has fairly good Graphical Processing Unit (GPU) support and is a fast-maturing framework [37].

2.2.7.4 OpenCV

OpenCV is an open source, cross-platform library that provides building blocks for computer vision experiments and applications. It provides high-level interfaces for capturing, processing, and presenting image data. For example, it abstracts details about camera hardware and array allocation. OpenCV is widely used in both academia and industry [38].

2.2.8 Mobile Application:

The main goal of the project is to create a cross-platform mobile application that would provide both object detection based on image classification and object localization using the phone camera. The application should be fast enough to provide real-time detection to maximize the usability of the application. To achieve these goals, the architecture should follow the Flutter architecture so that it can take advantage of the optimized system performance.

2.2.8.1 Flutter Framework

Flutter is an open source framework by Google for building beautiful, natively compiled, multi-platform applications from a single codebase [39].

Flutter is a cross-platform UI toolkit that allows apps to interact directly with underlying platform services while allowing code reuse across operating systems like iOS and Android. Its goal is to make it easy for developers to create high-performance programs that feel natural across platforms.

2.2.8.2 Dart Language

Dart is a client-optimized language for developing fast apps on any platform. Its goal is to offer the most productive programming language for multi-platform development, paired with a flexible execution runtime platform for app frameworks [40].

Flutter uses the Dart language to build the applications. Dart language was developed in 2011, and partly because of the rising popularity of Flutter, the language has developed faster in recent years than before. Dart is a strongly typed object-oriented language.

2.3 Literature Review

Many studies have been conducted in order to efficiently automated weeding. Some of these studies focus on removing weeds using thermal techniques, while others employ chemical herbicides, lasers, and motorized arms, and we summarize a few of them below:

2.3.1 Neural Network Based Weeding Robot For Crop And Weed Discrimination [41]:

The purpose of this article is for detecting weeds in between crops and drilling them with a blade connected to the robotics' tail. Despite the fact that this robot is self-contained, it does not require additional manpower. It reduces the amount of working time. This technique of weeding is environmentally friendly and does not harm the crop's organic structure.

2.3.2 Development of an Autonomous Electric Robot Implement for Intra-Row Weeding in Vineyards [42]:

Weeding inside rows is a time-consuming and difficult process. As a result, a rotating weeder implement for an auto electrical robot was created. It can be used to get rid of weeds in orchards and vineyards' intra-row areas.

2.3.3 Plant and Weed Identifier Robot as an Agroecological Tool Using Artificial Neural Networks for Image Identification" [43]:

A plant and weed identifying tool based on artificial deep neural networks was conceptualized, built, and trained for the purpose of weeding the inter-row space in crop fields in this study. A high-level design of the weeding robot is conceptualized and proposed as a solution to the problem of weed infestation in farming systems. Data collection, data pre-processing, training, and optimizing a neural network model are all part of the implementation process.

2.3.4 Automatic weed detection system and smart herbicide sprayer robot for corn fields" [44]:

The purpose of this study is to provide a new approach for weed identification and classification that can be used by autonomous weed control robots. Plants must be classified into crops and weeds based on their properties, which is accomplished using a machine vision algorithm. 73 corn field images were obtained and selected to evaluate the algorithm's performance, and an overall classification accuracy of 95.89 percent was achieved.

2.3.5 Integrating machine vision-based row guidance with GPS and compass-based routing to achieve autonomous navigation for a rice field weeding robot" [45]:

The goal of this study was to integrate GNSS, compass, and machine vision into a rice field weeding robot in order to achieve fully autonomous navigation for the weeding operation, the robots need to be exactly guided through the crop rows without damaging the rice plants, and they need to be able to detect the end of the crop row and make turns to change rows, also a camera installed at the front of the robot was used to extract the four immediate rows by a novel crop row detection algorithm, and these rows were used to determine a guideline for the robot to exactly maneuver it along the crop rows. The proposed system was found to be working well in low weed concentrations, with a 45.9 mm average deviation from the ideal path and a heading correction accuracy of less than 2.5° which may decrease as the weed concentration increased, but despite that, the robot was still able to navigate without causing major damage to the plants.

2.3.6 Robotic weeder can improve weed control options for specialty crops [46]:

Labor shortages and increasing manual weeding costs are affecting conventional and organic specialty crop farmers, also the agrochemical industry does not prioritize specialty crop herbicides, and many of these crops lack access to effective herbicides, and because new herbicides are expensive to manufacture and small in number, now is the time to invest more in robotic weeders, organic crops require more advanced weed control technology, so robotic weeders are promising new weed control tools for specialty crops, because they are cheaper to develop and, with fewer environmental and human health risks, are less regulated than herbicides.

2.3.7 Robotic weed control using automated weed and crop classification [47]:

In this paper, a non-overlapping multi-camera system is used to give the weed control system more flexibility while coping with indeterminate classification delays. The proposed modular weed control unit is designed, implemented, and tested using mechanical and chemical weeding equipment. To lead the equipment to achieve high-precision weed removal, a framework is being developed that performs naive Bayes filtering, 3D direct intra- and inter-camera visual tracking, and predictive control while integrating state-of-the-art crop/weed recognition algorithms.

2.3.8 Design and application of a weed damage model for laser-based weed control [48]:

Chemical and thermal weed control technologies in agriculture are both environmentally and energetically questionable. The utilization of laser technology appears to be a promising alternative. This study examines the effect of CO_2 laser radiation (10,600 nm) on three growth stages of two weed species, monocotyledonous: Echinochloa crus-galli and dicotyledonous: Amaranthus retroflexus, using three laser spot diameters, three laser spot positions, and six laser intensities. Two weeks after irradiation, the lethal effect of irradiation was demonstrated by a 90 % reduction in weed fresh mass compared to untreated plants. Weed-specific laser damage models were created and tested, mapping the laser application's probability of success (p success= 0.95). The results showed that lethality was greatest if treatment was performed at early growth stages with high intensity. After conducting studies on monocotyledonous and dicotyledonous plants, it was found that Monocotyledonous 2-leaf plants were destroyed at high energy levels, whereas 4-leaf plants were hard to kill. Dicotyledonous 2-leaf-plants were already damaged at moderate intensities. As a result, monocot damage needed higher minimum laser doses than dicot damage. The damage models developed here can be used with robotics and image processing to develop laser-based weed control.

Conclusion:

We came to the following conclusion after reviewing the studies: herbicides stay in the environment, damage non-target plants, and cause health impacts in animals and humans.

Furthermore, farmers are aware that conventional herbicides are becoming less effective as herbicide-resistant weeds evolve and spread. Mechanical weed management via tillage appears to be the only non-herbicide weeding approach available today, however it is energy-intensive, harmful to soil and roots, and increases erosion. Weed-control methods based on thermal effects, such as spot-flaming or electrical resistance heating, are also environmentally friendly, but they have large energy costs and are not widely used. As a result, we apply weed control using a laser source in our project, which can considerably reduce weed growth. Only the weeds' growth centers need to be treated with lasers, as this has an impact on the entire plan. It also eliminates the use of herbicides while increasing productivity and competitiveness. Our robot will eliminate the health risks and environmental negative effects associated with herbicide use.

2.4 System Hardware Options

In this section, we will compare and examine the different available hardware options that we will choose from in order to build the robot.

2.4.1 Smartphones vs Microcomputers

When we think about building a prototype of an intelligent weeding robot, typically we search for the most appropriate microcomputer available like JetsonNano, Raspberry pi, Tinker Board and etc.

Using this approach to design a robot system requires buying a microcomputer and sensors like a camera module, and to display the results in a convenient way to the user we also need to purchase a screen.

But what if we can dispense all of these costs and use our smartphones to prototype a robot?

All the sensors the engineer needs to design a robot are typically embedded in a smartphone. Moreover, current smartphones are powered with processors faster than 1.5 GHz, often multi-core and 4GB of RAM memory or more.

2.4.2 Microcontroller

In our prototype we chose to deploy the deep-learning model on a smartphone, but we still need a simple microcontroller to take the position of the detected object from the smartphone application via the serial communication through the Bluetooth module and to control the motors of wheels and arm.

A low-cost microcontroller with enough pins to connect with motor drivers and the Bluetooth module will be adequate for our purpose. In our case, the most two low-cost boards available and suitable in our application are Arduino NANO and Arduino UNO. We review the specifications of the two boards and a comparison between them in the following table 4:

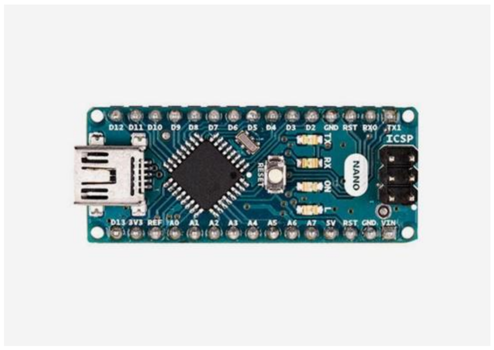
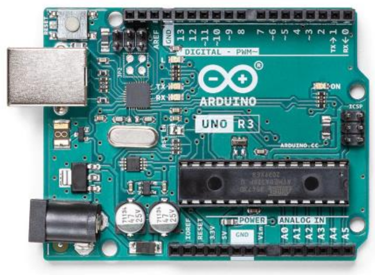
Microcontroller		
	✔ Arduino Nano 	**Arduino Uno** 
Processor	Atmega328P	Atmega328P
Input Voltage	5V / 7-12V	5V / 7-12 V
Speed of CPU	16 MHz	16 MHz
Analog I/O	8	6
Digital IO / PWM	14 / 6	14 / 6
EEPROM / SRAM [kB]	1 / 2	1 / 2
Flash	32	32
USART	1	1
Size	45×18mm	68.6×53.4mm

Table 4: Comparison between Arduino Nano and Arduino Uno

We see that Arduino Nano is similar to Arduino Uno board in most specifications, but the main difference between the two boards is size. Uno size is double to Nano board, so it takes more space on the system.

2.4.3 Actuators

2.4.3.1 Wheels and Arm Movement Motors options:

Motors		
Type	**Pros**	**Cons**
DC Motor	Cheaper than many other categories. Reduced sparking and electrical noise, so good for hazardous environments. Often have 2 or 3 wires.	More complex to control. To control position with a DC motor we need an encoder to give a feedback to the microcontroller.
✔ Stepper Motor	Has a similar characteristic to a DC motor, however we can command it in "steps" so we can do position control with no external feedback.	The primary advantage is also the disadvantage. In many high torque and operations where things change (like in a robot) it can skip steps leading to position error.
RC Servo	Small, easy to control motors that can be given position commands.	By default it has about 100 degrees of movement (each brand is different). They need to be modified for continuous rotation. Only for small torque operations.

Table 5: Comparison of Motors

We will use Stepper motors for wheels and arm movement, since we need to control the position of the laser in 2-dimensions, so we will use two stepper motors for wheels to control the location of laser in y-dimension, and the arm stepper motor to control the location of laser in x-dimension.

2.5 System Hardware Components

In this section, we will present the hardware components that we will use in this project, give a brief description, show the important specifications and explain the usage of each component.

2.5.1 Arduino Nano Microcontroller

Arduino Nano is a small size microcontroller based on Atmega328 processor, and designed by Arduino.cc. This board doesn't have any DC jack, and works with a Mini-B USB or we can connect the board to a power supply straightly via pins VIN and GND. The ATmega328 has 32 KB, (also with 2 KB used for the bootloader). The ATmega328 has 2 KB of SRAM and 1 KB of EEPROM [49].

Each of the 14 digital pins on the Nano can be used as an input or output, using pinMode(), digitalWrite(), and digitalRead() functions. They operate at 5 volts. Each pin can provide or receive a maximum of 40 mA. The following figure show the pon layout of Arduino Nano board:

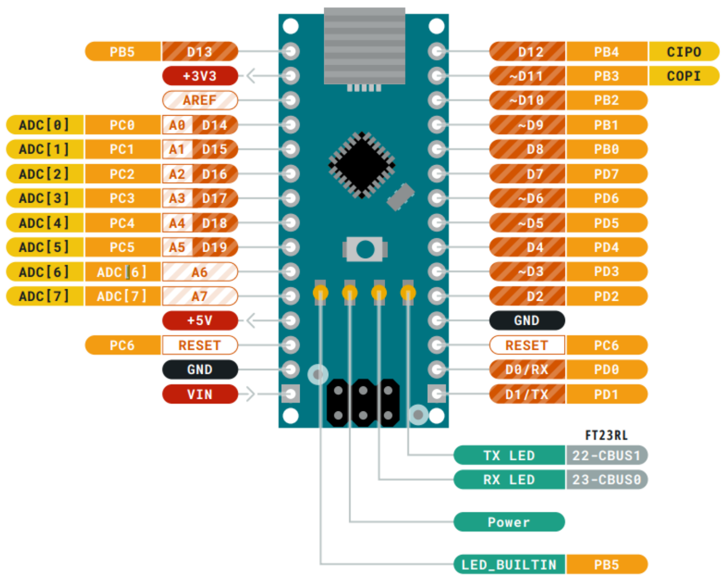


Figure 4: Arduino Nano pinout[65]

Communication:

In our system we need to create a communication between the smartphone and the microcontroller. The Arduino Nano has a number of facilities for communicating with a smartphone, a computer, another Arduino, or other microcontrollers. The ATmega328 provides UART TTL (5V) serial communication, which is available on digital pins 0 (RX) and 1 (TX).

An FTDI FT232RL on the board channels this serial communication over USB and the FTDI drivers (included with the Arduino software) provide a virtual com port to software on the computer. The Arduino software includes a serial monitor which allows simple textual data to be sent to and from the Arduino board. The RX and TX LEDs on the board will flash when data is being transmitted via the FTDI chip and USB connection to the computer. A **SoftwareSerial** library [50] allows for serial communication on any of the Nano's digital pins. The ATmega328 also supports I2C (TWI) and SPI communication.

2.5.2 Stepper Motor

After determining the position of the weed in the image, we need to reach the physical position of the weed in the real environment, so we need to choose this type of motors that give us the ability to control the position precisely. We will use two Nema17 Bi-polar stepper motors (FL42STH38-1684A) for the wheels; because this motor has a high output torque, high-precision positioning and small vibration. We can drive this type of stepper motors using EasyDriver motor drivers.

Specifications of FL42STH38-1684A: [51]

- Current per Phase: 1.68 A
- Rated Voltage: 2.8 V
- Holding Torque: 3.6 Kg.cm
- Detent Torque: 150 g.cm
- Step Angle: 1.8 DEG.

Figure 5: FL42STH38-1684A Stepper Motor

Also we need another stepper motor to control the position in x dimension, so we will use a unipolar stepper motor (SST42D2120) with the linear slide; this type of motor has the enough torque to handle the movement of the laser handler [52].

Specifications of SSTSST42D2120:

- Current per Phase: 1.2 A
- Voltage: 3.7 V
- Holding Torque: 3.2 Kg.cm
- Step Angle: 1.8 DEG.

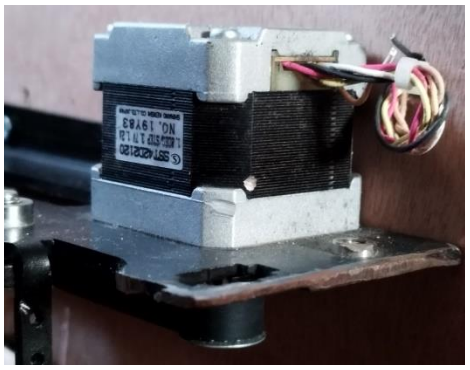


Figure 6: SSTSST42D2120 Stepper Motor

2.5.3 L298N Motor Driver

We will use the L298N motor driver to drive the unipolar stepper motor which is connected with the linear-slide that will carry the laser. L298N has two H-Bridges, each H-Bridge will drive one of the electromagnetic coils of a stepper motor. By energizing these electromagnetic coils in a specific sequence, the shaft of a stepper can be rotated forward or backward precisely in small steps.

The following are the most important specifications of the L298N[53]:

- Operating supply voltage up to 46V
- Low saturation voltage
- Maximum Driver current: 2A
- Has a heat sink for over temperature protection
- High Noise Immunity

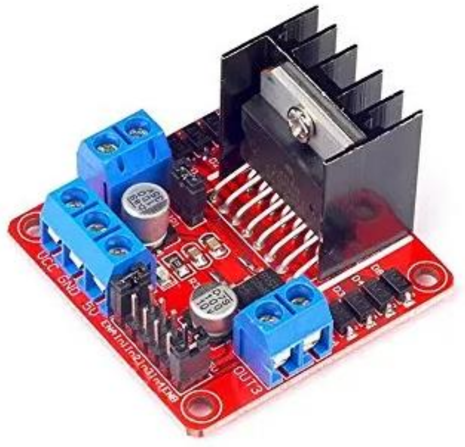


Figure 7: L298N motor driver

2.5.4 EasyDriver Motor Driver

EasyDriver is an IC enabled to drive bi-polar stepper motors [54]. We will use two EasyDriver stepper drivers to control the bi-polar stepper motors that are connected to the wheels. Each EasyDriver can drive up to about 750mA per phase of a bi-polar stepper motor.

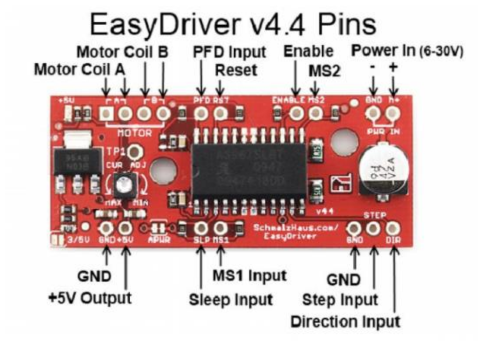


Figure 8: EasyDriver board [55]

2.5.5 HC-06 Bluetooth Module

To create a serial communication between the microcontroller and the smartphone, we will use an HC-06 module, which is a Bluetooth module designed to establish short range wireless communication between two devices.

This module has the following specifications [56]:

- Sensitivity (Bit error rate) can reach -80dBm.
- Has a build-in 2.4GHz antenna
- Low power consumption
- Has high-performance wireless transceiver system
- Low Cost

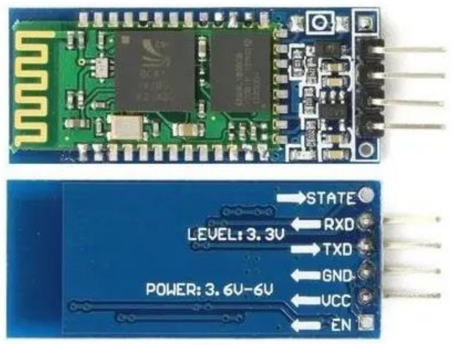


Figure 9: HC-06 Bluetooth module

2.6 System Constraints

1. Climate change, rainfall, strong winds may harm the robot and have an impact on the robot’s functions.
2. Intertwining weeds with crops, the classification process may be difficult, and the removal process may affect the crops.
3. The robot's battery has to be charged all the time.
4. To avoid crop damage, the system must react fast enough.
5. Crops and weeds might grow to be higher than the robot.
6. Time constraint; we have to end the work on our project before the deadline.

Chapter 3 System Design

3.1 Overview

In this chapter, we will discuss the overall design of the Weeding Robot system, and how its components are integrated. We will start by showing the abstract block diagram of the system, and next the detailed conceptual description of the system, and then we will show the system flowchart and pseudo code. In the end we will explain the software and hardware design of the system.

3.2 Brief Description of the system

Our Weeding Robot system aims to detect and identify weeds, using a deep learning model that will be deployed on a mobile application. The Weeding Robot is a car-like machine, which moves in one direction in the crops row.

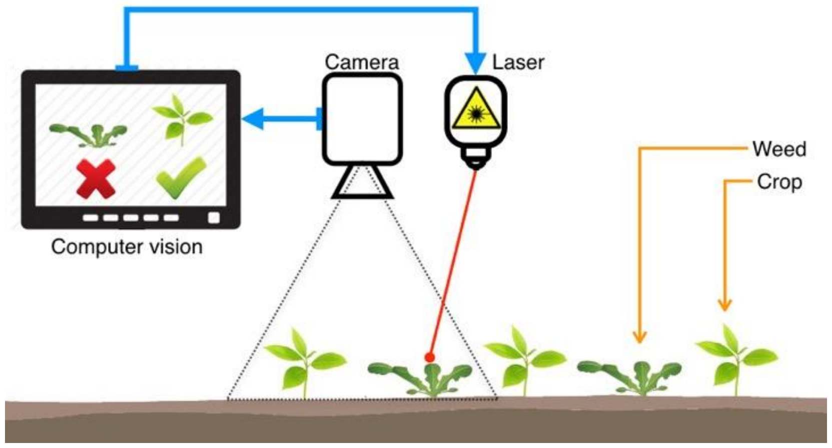


Figure 10: Laser-based weeding, general view of the system [57]

As represented in figure 10, after identifying the weed plants, a laser beam will be used to eliminate or to seriously damage the weeds. In this way, valuable crops can grow without the competition from weed and have higher yields, since the available nutrients do not have to be shared.

3.3 Detailed Hardware Design

3.3.1 Detailed conceptual description of the system

The system we introduce is a Weeding Robot that is designed in the form of a prototype that is operated using an Android-based smartphone through a flutter application that has been installed. Figure 11 shows the high-level hardware design of the system. As presented below, the weeding robot system collects images using the smartphone camera while moving through the crop row. The smartphone processor will process the images by doing some preprocessing operations on each frame and then detect if there are objects in the image using the deep learning model that is deployed on the smartphone to make the appropriate decision. If there are no objects in the image classified as weeds, the smartphone will not send any data about the object position to the Arduino, so in this case the microcontroller will continue sending signals to the navigation motors to keep wheels moving. If there an object in a frame classified as a weed, then the smartphone will send the position of weed to the microcontroller, which in turn will send signals to wheels and arm motors to move precisely above the weed location, then the microcontroller sends signal to turn on the laser for specific amount of time, after that, the microcontroller sends signals to the navigation motors to continue movement.

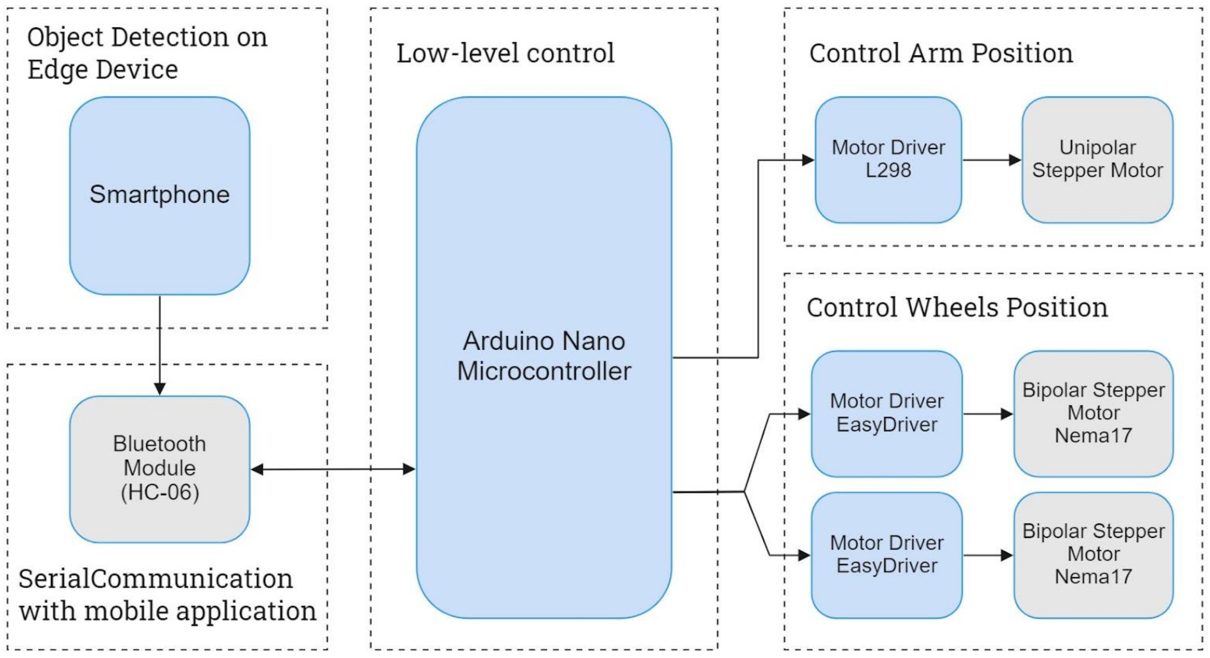


Figure 11: High-level hardware design block diagram.

3.3.2 System Schematic

Figure 12 demonstrates the schematic diagram of the weeding robot prototype. It displays the various hardware components of the system and their interconnections.

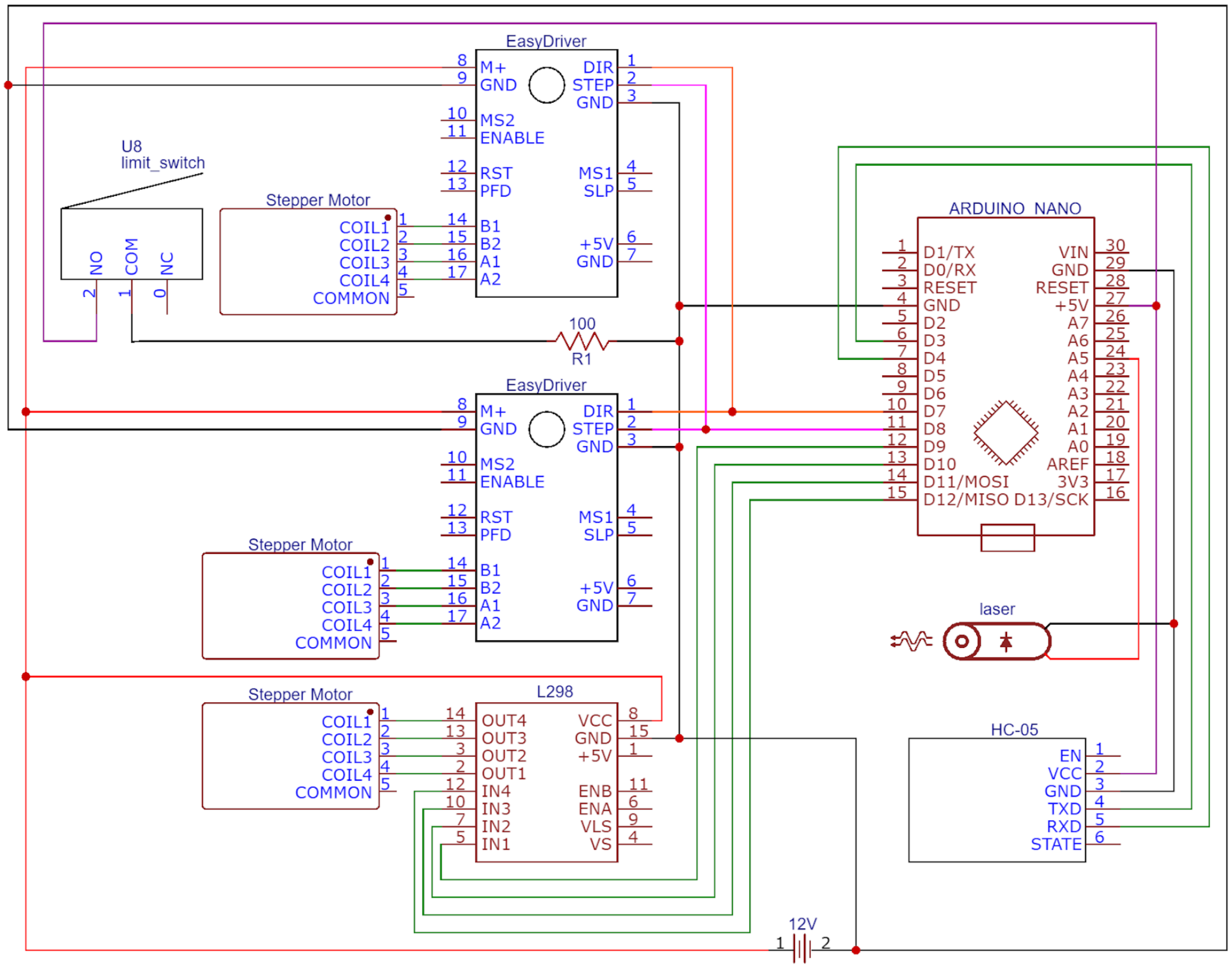


Figure 12: A Schematic Diagram for the system.

The Arduino Nano connects the three motor drivers via the General-Purpose Input/output (GPIO) ports. The Arduino microcontroller controls the Unipolar stepper motor that is responsible for arm movement by sending signals to the four input pins of L298N H-Bridge, where the L298N connects with the Arduino board on pins D9, D1, D11 and D12.

The two Nema17 Bipolar stepper motors are connected to two EasyDriver motor controller chips which are controlled by the Arduino Nano microcontroller to supply voltage levels of 0 or 5V at the ports D7 and D8, which we control the direction of rotation on the D7 pin and we control the number of steps on pin 8. The three motor drivers are directly powered by the 12V battery.

The HC-06 Bluetooth module connects to the microcontroller, to enable the serial communication with the smartphone, where the pins TXD and RXD of the Bluetooth module are connected with the pins D3 and D4 on the Arduino to send and receive data. Arduino controls the laser light on pin A5 by sending HIGH digital signals for a specific time when the laser is reached the weed position. A limit switch is connected with the Arduino, where it sends an interrupt signal to the Arduino on the digital pin 2 when the limit switch is in the RISING state.

3.4 Detailed Software Design

3.4.1 Description of the software design

We will train our deep learning model on a dataset of weed, and we will use the YOLO algorithm. YOLO algorithm provide a real-time object detection uses Neural Network, it works by bounding a box specifying object location, each bounding box can be described using four descriptors: center of the box (bx, by), width (bw), height (bh) and value c corresponding to the class of an object then dividing the image into N grids, each having an equal dimensional region of SxS. Each of these N grids is responsible for the detection and localization of the object it contains, and for each grid cell, the label y will be an dimensional vector containing the pc defines whether an object is present in the grid or not, and bx, by, bh, bw specify the bounding box if there is an object, and finally the total number of classes, and in our system we have two classes, weed and plant. After that if the system detects a weed then a laser beam is directed on it.

3.4.2 Flowchart

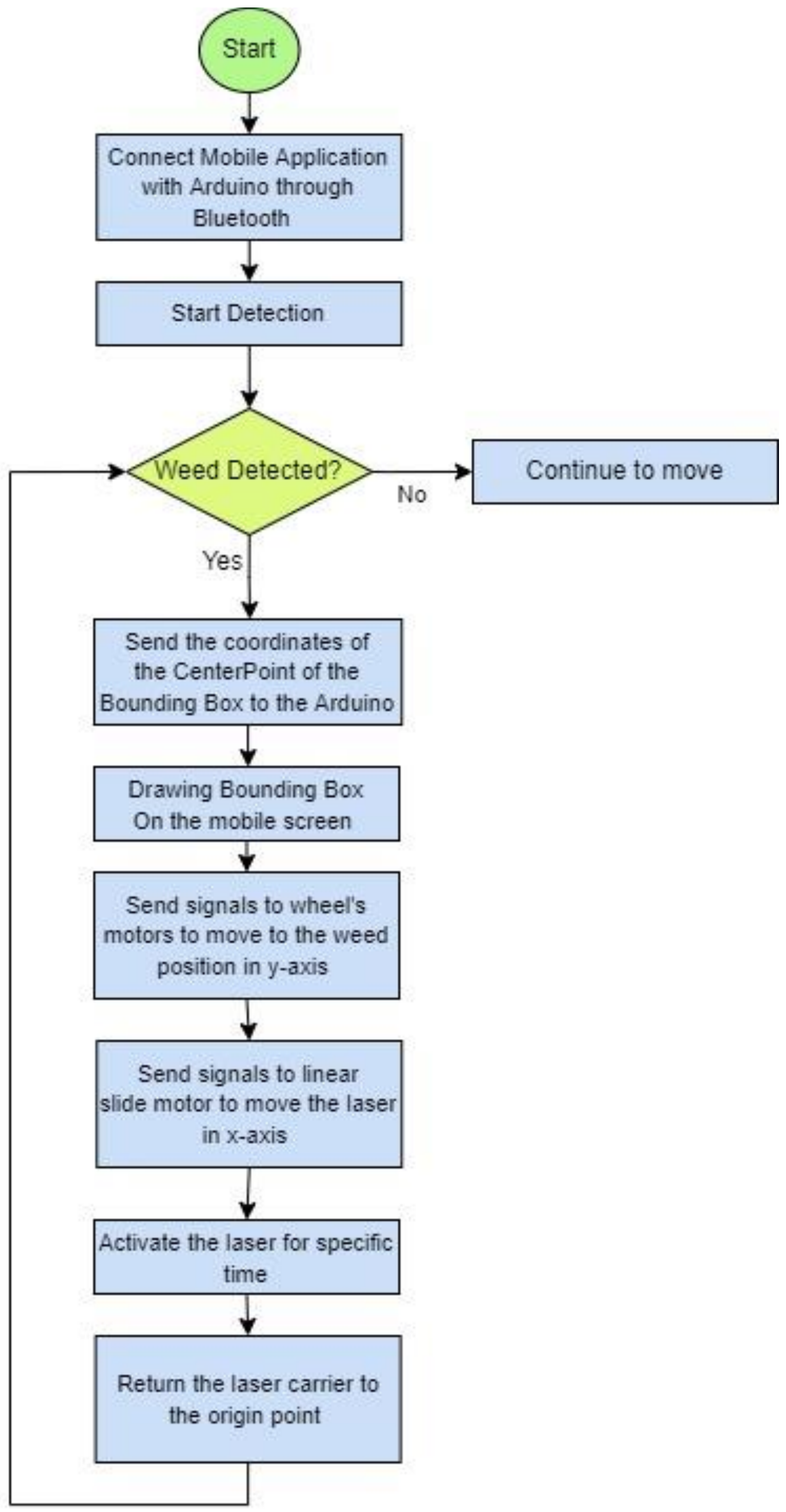

Start
Connect Mobile Application with Arduino through Bluetooth
Start Detection
Weed Detected?
No
Continue to move
Yes
Send the coordinates of the CenterPoint of the Bounding Box to the Arduino
Drawing Bounding Box On the mobile screen
Send signals to wheel's motors to move to the weed position in y-axis
Send signals to linear slide motor to move the laser in x-axis
Activate the laser for specific time
Return the laser carrier to the origin point

Figure 13: System Flowchart.

3.4.3 Pseudo Code

The following lines have a description of the pseudo code used in the system:

```
Start
Connect Mobile Application with Arduino through Bluetooth
StartDetection
REPEAT
if weed is detected
        Send the coordinates of the CenterPoint of the Bounding Box to the Arduino
        Drawing Bounding Box On the mobile screen
        Send signals to wheel's motors to move to the weed position in y-axis
        Send signals to linear slide motor to move the laser in x-axis
        Activate the laser for specific time
        Return the laser carrier to the origin point
else
        Continue to move
UNTIL power is off.
end
```

Chapter 4 System Implementation

4.1 Overview

This chapter describes the implementation part of the system. It presents the different components and the assembly of the different electronics and mechanical parts of the system, as well as the interconnection and the integration of the system components.

4.2 System Hardware Assembly

This section describes the electronics and mechanical parts assembly and arrangement in the robot system. The following points explain the hardware assembly process:

- Starting with the X Axis Linear Rail Kit, we connected it with a stepper motor to control the location of the laser in the x-axis.

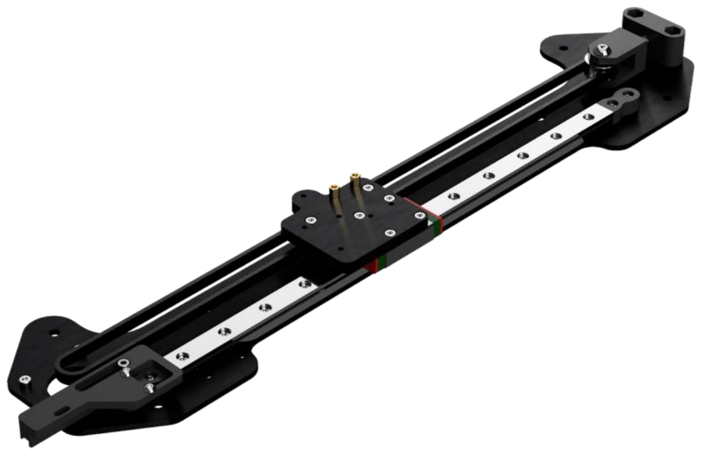


Figure 14: X-Axis Linear Rail Kit

And then, we installed the kit on the wooden body of the robot (as shown in figure 15) to allow the laser to move freely and appropriately over the crop row.

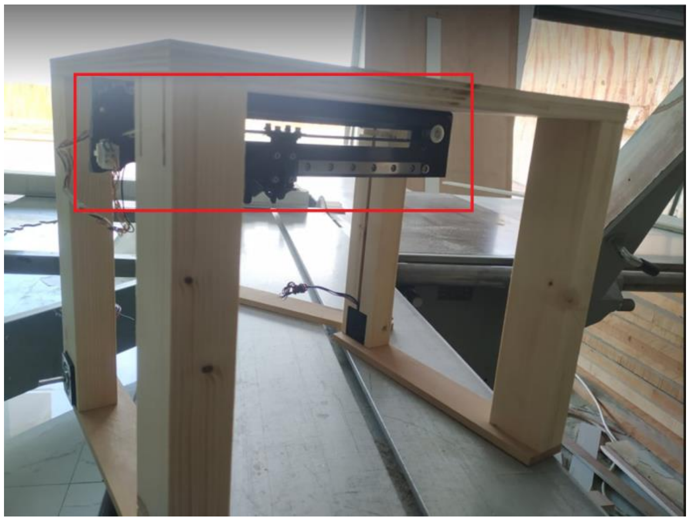


Figure 15: Linear Rail Kit Installation

- For the wheel motors, we design a carrier for them in the wood legs of the robot, which allows us to insert the stepper motors of the wheels in a good way that allows them to carry the robot body and allows the wheels to rotate freely. as shown in figure 16:

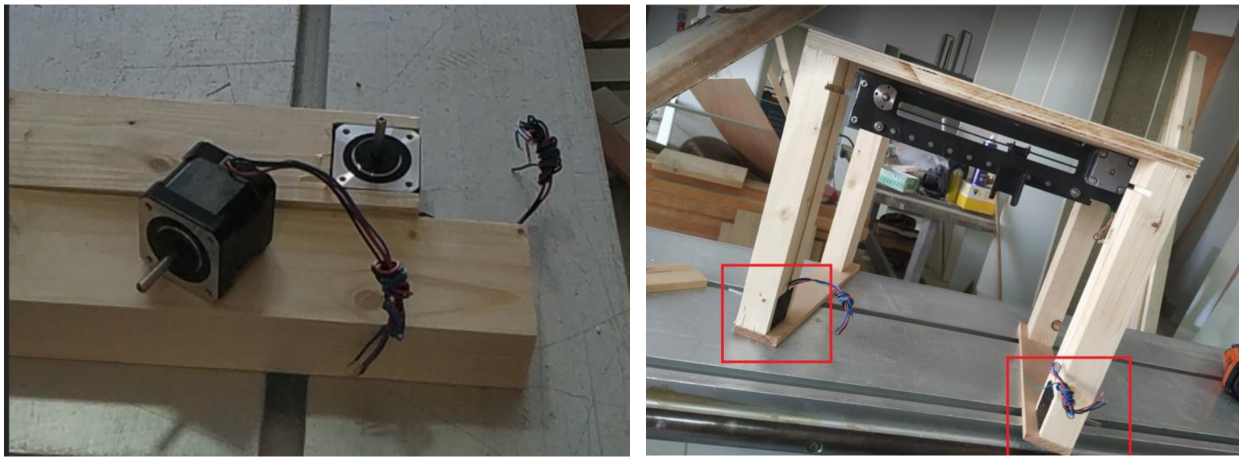


Figure 16: Wheels motors

- We installed the Arduino Nano microcontroller, HC-06 Bluetooth module and the two EasyDriver motor drivers on a breadboard on the top of the robot body. We also installed the L298n motor driver on the top of the robot as in figure 17:

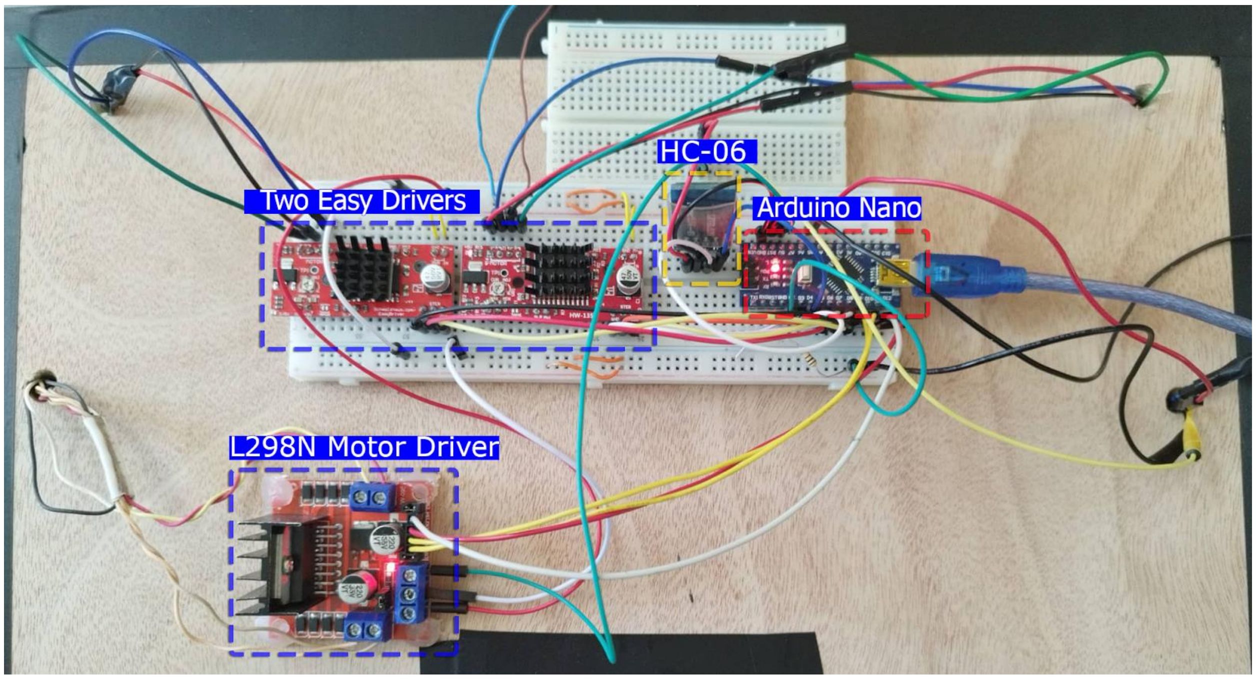


Figure 17: chips installation

- Fixed place is created on the top of robot wooden body to hold the smartphone, as shown in figure 18:

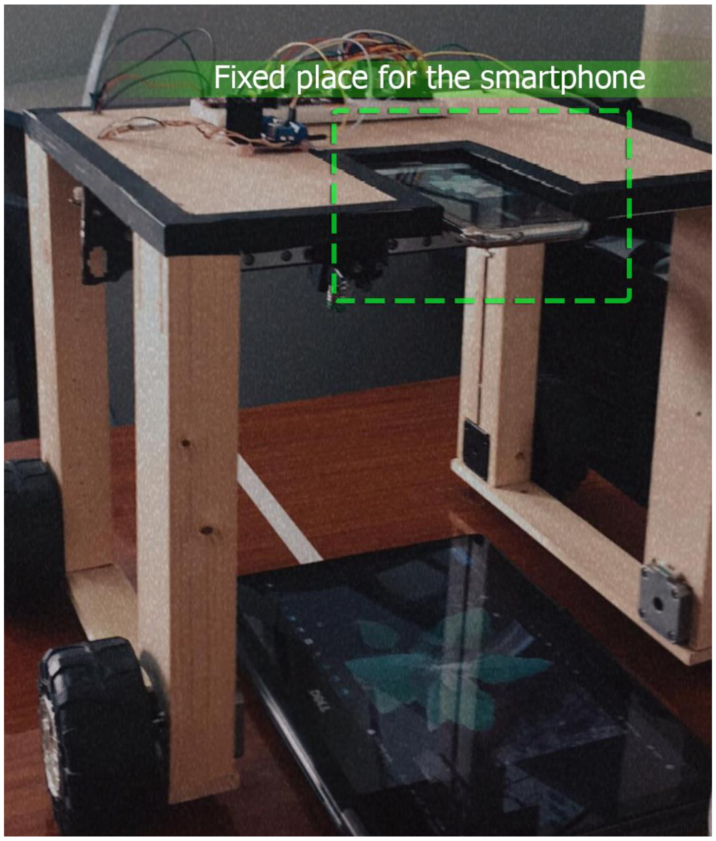


Figure 18: Fixed place for the smartphone

Where we should keep the position of the camera in a fixed place to do the mapping between the coordinates on the smartphone screen from the camera and the coordinates in the real environment. As well, the camera should be always in a fixed place on the robot body to track the position and eliminate weeds correctly.

- We used a laser light that can be controlled by the microcontroller and attached it to the laser carrier.
- Finally, we Added a limit switch to the linear rail kit, because we found an error rate that accumulated with time when moving many times in the x-axis, so we return the laser carrier to the 0 point in the x-axis after each frame detection.

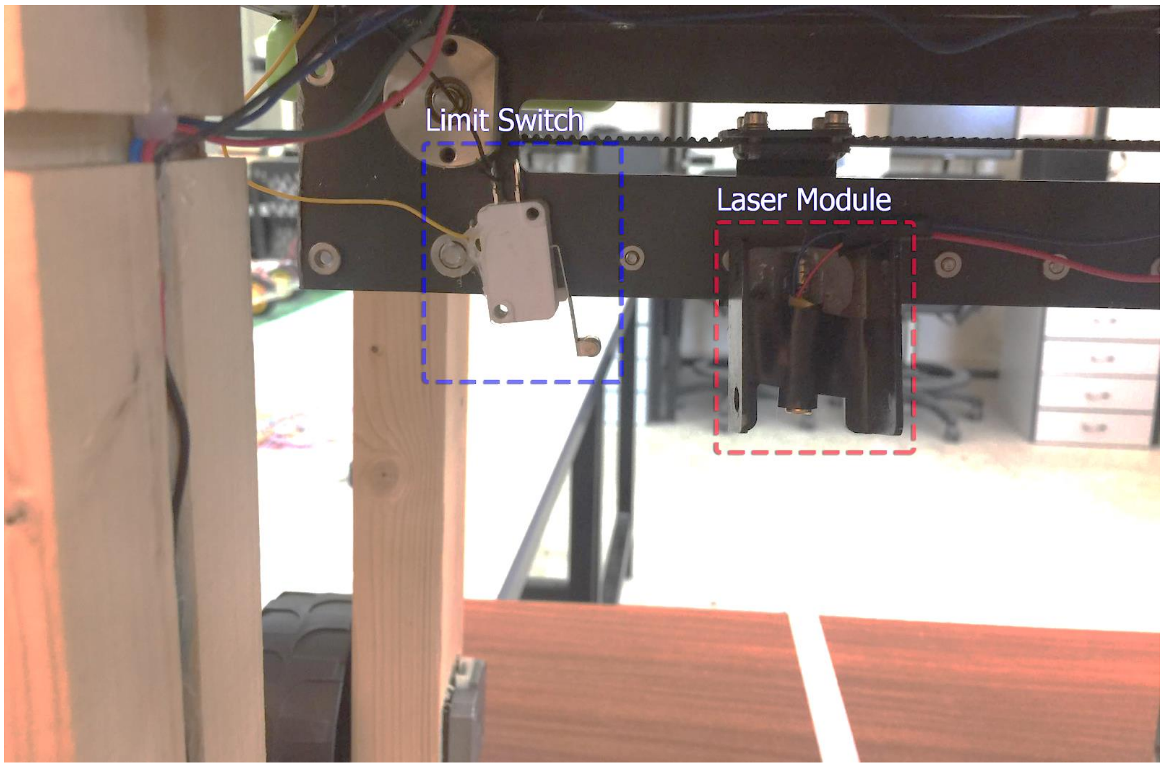


Figure 19: Laser and Limit Switch

4.3 System Software

This section goes through the specifics of how the system's various software components are implemented, we have three main components which are: Dataset, Model And Mobile application, in the dataset part we gathered the images and generate bounding boxes annotations, in the model part we trained the dataset with various models and converted them to tflite, and in the mobile application, we connected the app with the Arduino via the Bluetooth and the model was deployed on it. And finally we have the real time object detection application.

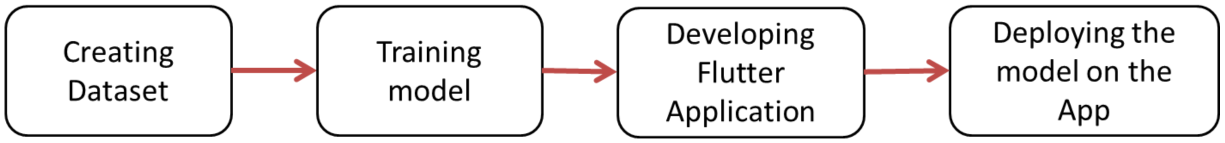

Creating Dataset
Training model
Developing Flutter Application
Deploying the model on the App

Figure 20: System Software Implementation

4.3.1 Dataset

Before diving into the software details, it is important to talk about the data engineering process and how we gathered, annotated and preprocessed the data to train the deep learning model.

4.3.1.1 Value of the Dataset

This dataset is a collection of images of weeds at different stages of development. The images are annotated with weed bounding boxes. These data can benefit research institutes working on the automatic detection of weeds as well as commercial applications exploiting Deep Learning models for real-time detection. They can also benefit any research in computer vision and artificial intelligence that aims at developing new weed detection and location algorithms.

4.3.1.2 Dataset Description

The dataset is composed of 1400 JPEG images. Each image is uniquely identified by its file name. There was a number of datasets for such types of weed available online, but it was limited to a foreign countries. Therefore, we decided to search and ask for the most common weed types in Palestine. Five types have been chosen: Malva sylvestris, Senecio vulgaris, Silybum marianum, Chenopodium album and Urtica urens.

4.3.1.3 Dataset gathering and preprocessing

4.3.1.3.1 Data Gathering

GBIF "Global Biodiversity Information Facility" website was used to gather the images of the named weeds, this website is free and open access to biodiversity data [58]. Luckily, there was thousands of high-quality images, the images were gathered with some consideration, such as the upper perspective, different light options and different backgrounds.

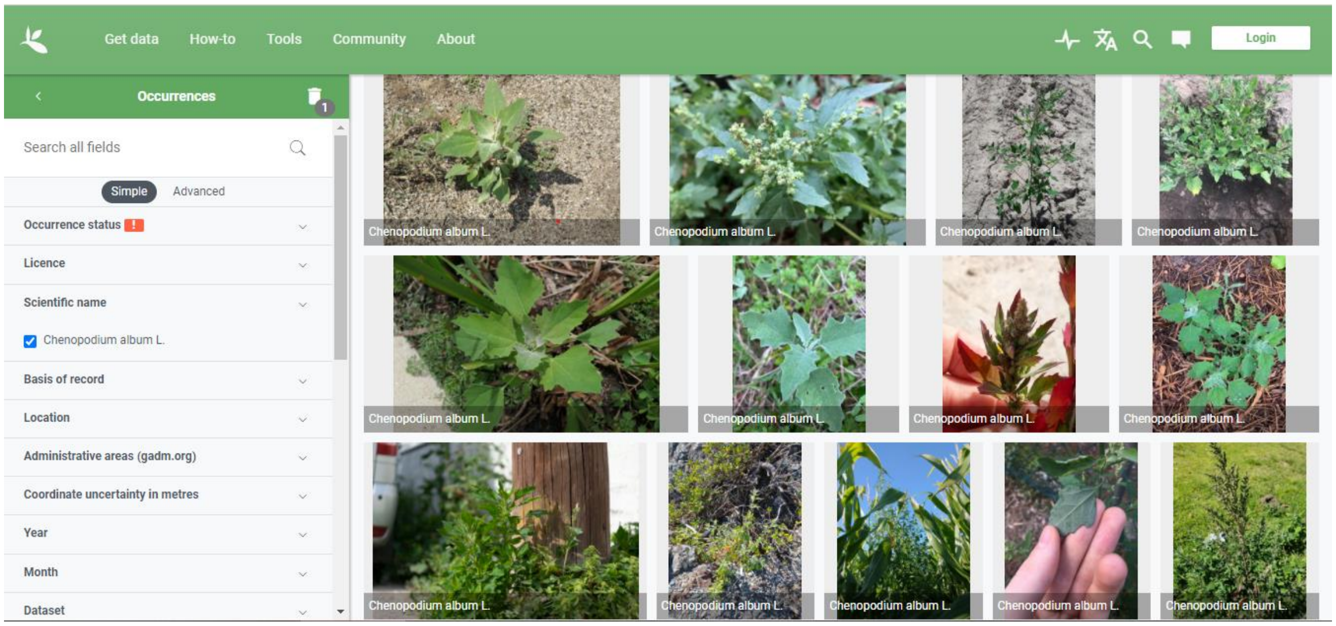


Figure 21: gbif

4.3.1.3.2 Bounding Boxes Annotations

Makesense.ai is a tool for image annotation available online, which eases the process of labeling images with bounding boxes. The tool enables the user to easily draw bounding boxes and annotate each box with a predefined label. There is no need for any setup or installation; it supports multiple label types such as rects, lines, points and polygons, also its support output file format like YOLO, VOC xml and CSV [59].

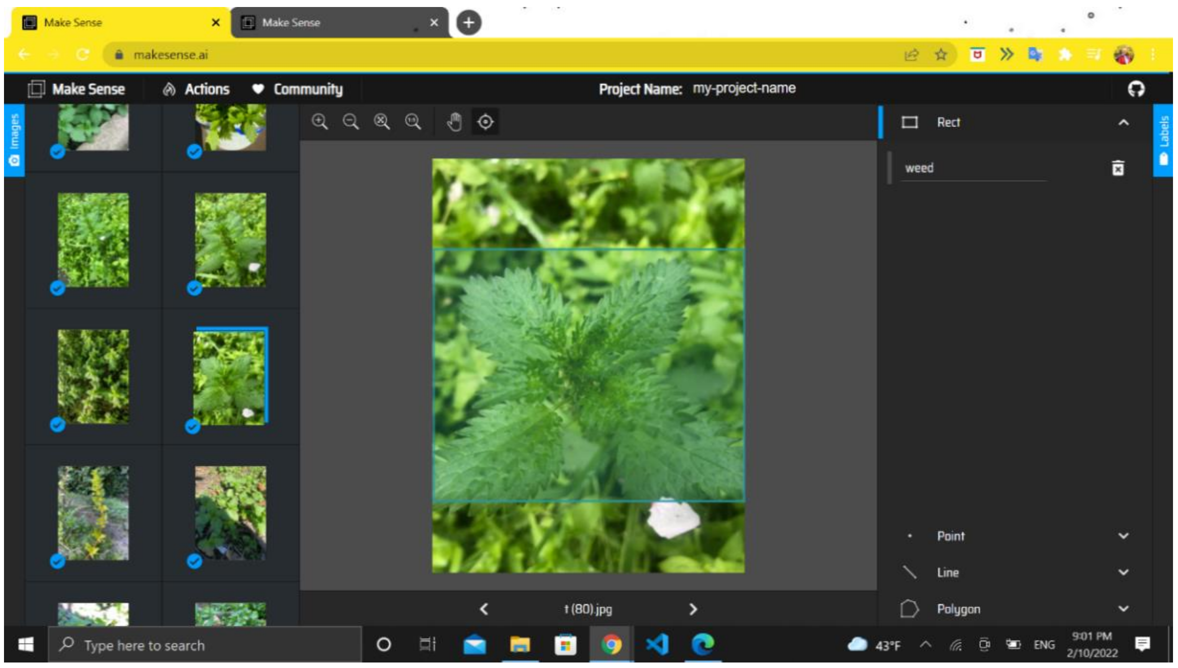
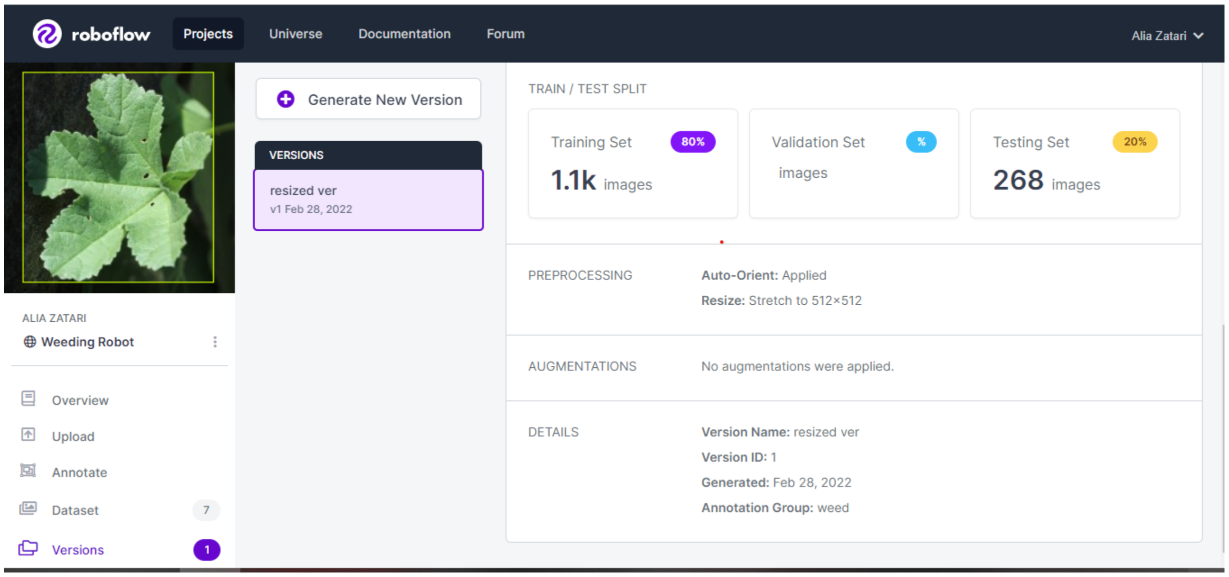


Figure 22: Make Sense.

4.3.1.3.3 Preprocessing

Roboflow empowers developers to build their own computer vision applications. It provides all of the tools needed to convert raw images into a custom trained computer vision model and deploy it for use in applications. Today, Roboflow supports object detection and classification models. In our case, we used this tool to preprocess our images dataset, Resize changes your images size and, optionally, scale to a desired set of dimensions. Annotations are adjusted proportionally.



TRAIN / TEST SPLIT

Training Set 80%	Validation Set %	Testing Set 20%
1.1k images	images	268 images

PREPROCESSING
Auto-Orient: Applied
Resize: Stretch to 512×512

AUGMENTATIONS
No augmentations were applied.

DETAILS
Version Name: resized ver
Version ID: 1
Generated: Feb 28, 2022
Annotation Group: weed

Figure 23: Roboflow.

4.3.2 Choice of base Models

Since Deep CNNs require intensive training, networks like SSD, R-CNN, and YOLO were chosen as detection frameworks for the CNN to be expanded and fine-tuned. The choice of the base model for the object detection algorithms among these models is highly influenced by the models speed versus performance ratio, also the speed and the accuracy varies heavily between the different models as well as the mAP between the networks.

4.3.2.1 Model Training

Training the models requires a computer with a reasonably powerful GPU. The use of a computer with a GPU is not necessary as the training is done via Google Colab. Google Colab provides an online environment used for Machine Learning. Using Google Colab, the training was done online via an external high-performance GPU. This choice of Machine Learning environment also makes the training faster than with a few years old GPU.

In our case, we chose to try different models, which are efficientdet_lite, YOLOv4, YOLOv4-tiny, and YOLOv5 to decide which one is suitable for our mobile application. After the trial of deploying the trained models on the mobile application, the choice of the model was dependent on the Flutter Dependencies version so YOLOv4 and YOLOv4Tiny successfully worked.

Training YOLOv4-Tiny is done via Google Colab. First we cloned the Darknet git repository onto the Colab Notebook. Darknet is an open source neural network framework written in C and CUDA. It is fast, easy to install, and supports CPU and GPU computation.

Second we created the necessary files for the training like *obj.name* file contains the name of classes and *obj.data* file has the number of classes, the path to *obj.names* file and the path to *train.txt* and *test.txt* files described in fourth step. Third, we downloaded the *yolov4-tiny-custom.cfg* file from the *darknet/cfg* directory and made some changes to it. Fourth we ran the *process.py* python script (Appendix A) to divide all image files into 2 files, *train.txt* and *test.txt* where the train.txt file has paths to 90% of the images and test.txt has paths to 10% of the images. After that we uploaded all these files to Google Drive and linked it to Google Colab.

We used transfer learning (2.2.3.2.4) so instead of training a model from scratch, we use pre-trained YOLOv4-tiny weights which have been trained up to 29 convolutional layers. Next step was to train the model and it took a few hours to finish, then we tested the model by running our custom detector on some images, the results shown in section 5.3.2 (Appendix B)[60]. Final step was converting the weights file into a pb file which is a protocol buffer file, then converted the pb file to TFLite format to be used on lower-performance mobile devices and edge devices such as a raspberry pi. [61].

4.3.3 Mobile Application Development

A mobile application was developed to research the possibilities of using Flutter in mobile real-time object detection, Flutter was chosen for this project because it is best suited for cross-platform systems that require rapid calculations as real-time object detection needs. The most important part of application functionality is to connect the camera to the deep learning model.

4.3.3.1 Bluetooth Module

This module is responsible for the connection with the Arduino, it sends the center point of the weed location to the Arduino microcontroller.

Our application starts with the Bluetooth page, in the first page we enable the user to power on the Bluetooth and show the available devices to connect with.

The following list summarizes the expected responsibilities:

- Listing paired devices to communicate.
- Sending and receiving data.

4.3.3.2 Camera Module

The main function of this app is created in such a way that all the cameras are initialized right after the Bluetooth connection. Then, the camera descriptions are passed on to all the classes that require it.

The following list summarizes the expected responsibilities:

1. Initialize and manage the image stream from the camera
2. Provide an interface for reading the image frame
3. Dispose the stream after closing the application

This model contains the following functionalities:

- initStateAsync()

Here the Camera initialization is created , then we create an instance of classifier to load model and labels and set the Initial predicting to false.

- initializeCamera()

Initializes the camera by setting [cameraController] and changing the raw input image size to be suitable with the mobile screen size.

- onLatestImageAvailable(CameraImage cameraImage)

This function is created to receive each frame [CameraImage], pass results to HomeView and set predicting to false to allow new frames.

4.3.3.3 Object Detection Model Deployment

This is the part where we deploy the object detection model. The first step was adding the model files (detect.tflite & label.txt) to the assets folder, and then we update the *pubspec.yaml* and add the following lines:

```
assets:

 - assets/detect.tflite

 - assets/label.txt

dependencies:

    tflite_flutter:  ^0.9.0

    tflite_flutter_helper:  ^0.2.1
```

tflite_flutter is a plugin that provides a flexible and fast solution for accessing TensorFlow Lite interpreter and performing inference and *tflite_flutter_helper* is a library brings TFLite Support Library and TFLite Support Task Library to flutter and helps to develop ML and deploy TFLite models onto mobile devices quickly without compromising on performance, the *camera* package for controlling the camera and the *image* is a library providing the ability to load, save and manipulate images in a variety of different file formats.

The following list summarizes the expected responsibilities:

1. Loading the model and the labels.
2. Pre-processing the input image.
3. Detect the objects.

This model contains the following functionalities:

- loadModel({Interpreter interpreter}) : Loads interpreter from asset.
- loadLabels({List<String> labels}) : Loads labels from assets.
- getProcessedImage(TensorImage inputImage) : Pre-process the input image.
- predict(imageLib.Image image) : Runs object detection on the input image and returns recognitions.

4.3.3.4 Bounding Boxes Generation

A bounding box is an imaginary rectangle that serves as a point of reference for object detection and creates a collision box for that object.

The following list summarizes the expected responsibilities:

1. Represents the recognition output from the model.
2. Return a bounding box for the detected weed.

This model contains the following functionalities:

- renderLocation()

Returns a bounding box rectangle corresponding to the displayed image on screen, this is the actual location where the rectangle is rendered on the screen.

4.3.3.5 User Interfaces

This section will introduce all the interfaces that were designed in the application. There are five main pages in our application. The first page gives a brief idea about the application, to get started with a Get Started button will navigate to the next screen.

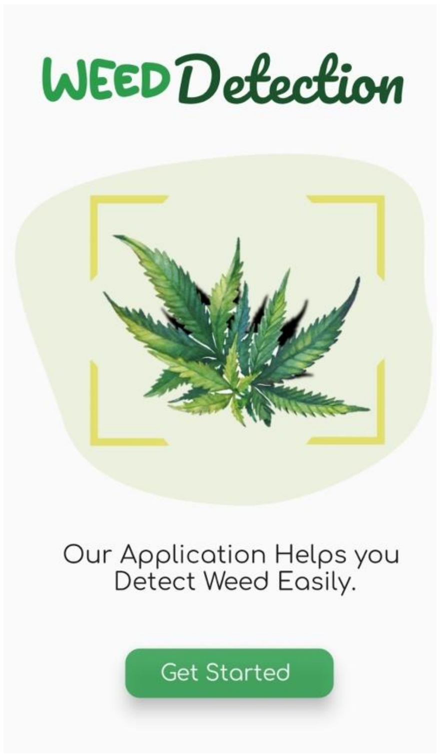
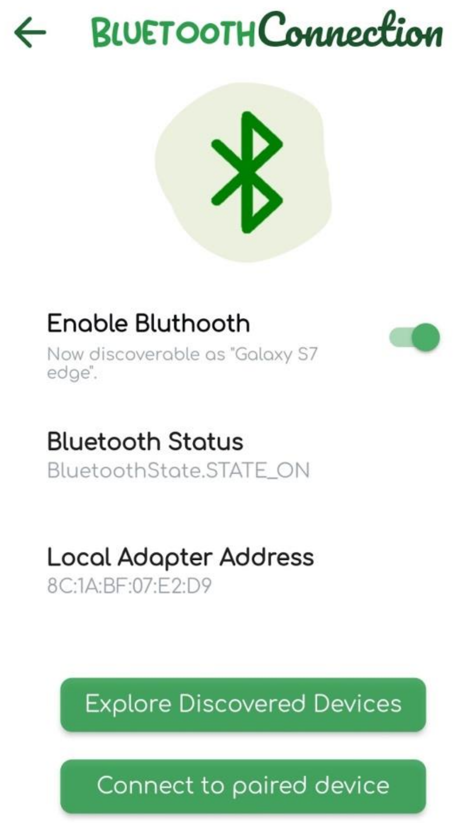


Figure 24: Main Interface

The second page for controlling the Bluetooth connection, it allows you to enable and disable Bluetooth and shows the name of the discovered devices, it's show the Bluetooth status, local adapter address, and ending with two buttons one for exploring the discovered devices, and the other one for connecting to paired devices.



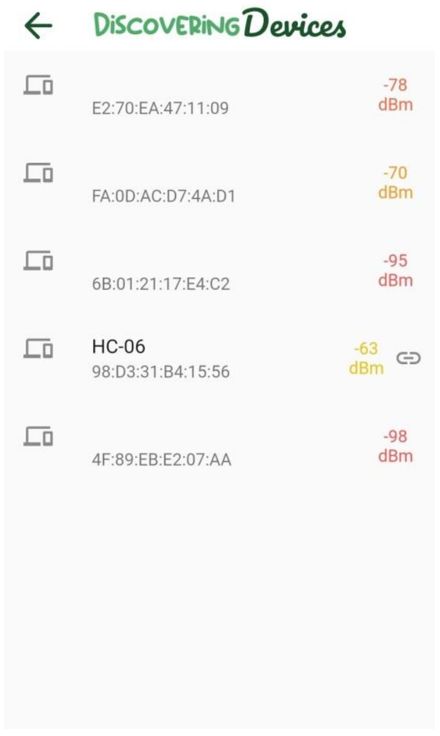
Enable Bluthooth
Now discoverable as "Galaxy S7 edge".

Bluetooth Status
BluetoothState.STATE_ON

Local Adapter Address
8C:1A:BF:07:E2:D9

Figure 25: Bluetooth Connection Interface

Clicking on the first button (Explore Discovered Devices) navigates to the third page, in this page a list of discovering devices appears.



E2:70:EA:47:11:09 -78 dBm

FA:0D:AC:D7:4A:D1 -70 dBm

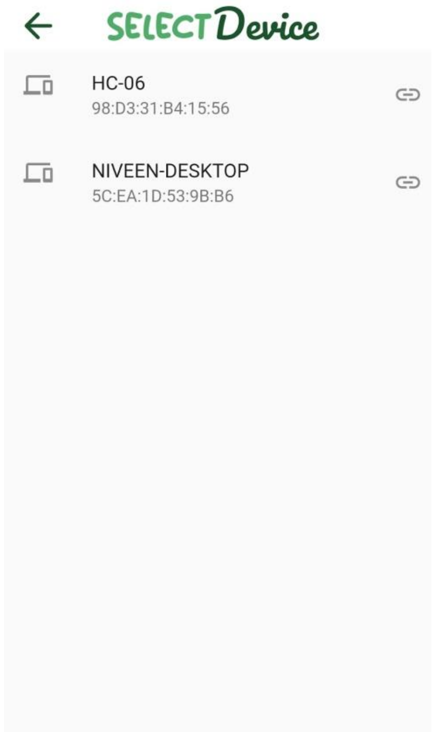
6B:01:21:17:E4:C2 -95 dBm

HC-06
98:D3:31:B4:15:56 -63 dBm

4F:89:EB:E2:07:AA -98 dBm

Figure 26: Discovering Devices Interface

Clicking on the second button (Connect to Paired Device) navigates to the fourth page; this page enables you to select the device you want to connect with.



HC-06
98:D3:31:B4:15:56

NIVEEN-DESKTOP
5C:EA:1D:53:9B:B6

Figure 27: Select Device Interface

After selecting the device, it will navigate you to the last page, which is camera streaming, in this page it will detect the weeds and draw bounding boxes.

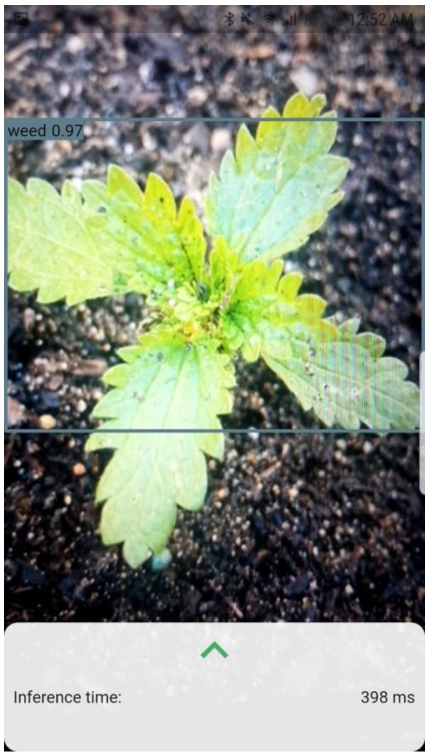


Figure 28: Camera Streaming Interface

4.3.4 Arduino Software Implementation

We used Arduino IDE to write the code and upload the sketch on the Arduino microcontroller. The code of the Arduino composed of the following components:

4.3.4.1 Communication Code

This code is responsible for creating the connection between the smartphone and the Arduino using the HC-06 Bluetooth module. In our application, the Arduino receives data from the mobile application. Before writing the communication code we should include the SoftwareSerial library.

The communication code contains the following functionalities:

- readPointString():

  This function use BTSerial.read() that provided by the SoftwareSerial library to read data received, were BTSerial is an instance we created for the serial communication on digital pins 3 & 4, read() function reads only one byte at a time, this means we can read only one character at a time. readPointString() can be summarized in the following points:

    - Convert receive byte from ASCII code to char
    - Check if it is an end character
    - Store the received characters until end character as one string represents the coordinates of received point.
    - After separating the string using getValue(), it converts each string coordinate to integer data type.

- getValue():

  This function takes the string point and the separator character as parameters and returns two string coordinates, that is, one of the two strings represents the x coordinate and the other represents y coordinate.

4.3.4.2 Motion Code

To write a code that controls the position of the robot and the laser, we had first calculated the number of steps that let the robot move a distance of centimeter, and also the number of steps that let the laser move one centimeter. After we found the corresponding number of steps for each value in centimeters, we wrote the motion code using Stepper library and we used the function step() to instruct the motor to rotate its shaft the desired number of steps to reach the required position.

The Motion code contains the following functionalities:

- moveToWeedPosition():
    - Accepts two integer parameters: x & y coordinates.
    - Instruct the motors to rotate the desired number of steps to reach the weed position.
- stopMotion():
    - called when an interrupt signal occurs on pin 2
    - Update the speed of wheels motors to 0 rpm.

4.3.4.3 Timer Code

To instruct the laser being HIGH for an amount of time after the laser light is positioned on the weed. This timer code contains the following functionalities:

- igniteLaser():
    - uses the millis() to indicate the start time before lighting the laser
    - Loops on HIGH state of the laser until the intended amount of time is reached.

4.4 System Integration & Location Mapping

Our main purpose of the whole system is to give the weeding robot the ability to detect weed and then direct the laser precisely on the weed to remove it. This purpose requires us to integrate all the system parts and write a function that can convert the position of the weed from pixels to centimeters to be able to let the motors rotate their shafts the required number of steps to reach the goal position.

The Weeding Robot system has two main parts that need to communicate with each other through a protocol; these two parts are the Arduino Nano microcontroller and the smartphone.

The smartphone used to run higher-level perception: the deep learning model (yolov4-tiny) is deployed on the smartphone to work as an edge device and to do the classification and localization of the objects in real time without connecting to the internet. The Arduino receives data and controls the actuators to do the appropriate action after the decision is done on the smartphone. The following subsections will detail the software stack of the system, how the integration was done, and how we mapped the object location to remove the weed precisely using the laser beam.

4.4.1 Software Stack

As we mentioned before, we can divide our system into two parts, so, the software stack of our system consists of two main components, illustrated in figure 29:

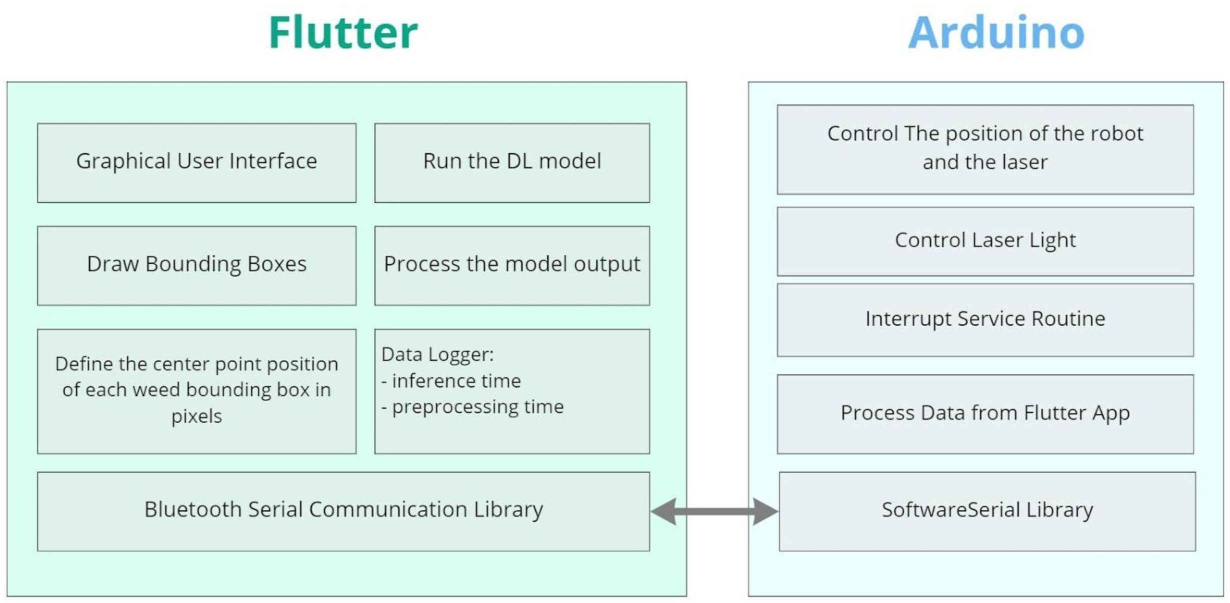

Flutter
Arduino
Graphical User Interface
Run the DL model
Draw Bounding Boxes
Process the model output
Define the center point position of each weed bounding box in pixels
Data Logger:
- inference time
- preprocessing time
Bluetooth Serial Communication Library
Control The position of the robot and the laser
Control Laser Light
Interrupt Service Routine
Process Data from Flutter App
SoftwareSerial Library

Figure 29: Communication between Flutter Application and Arduino

The first component is a Flutter application that's run on the smartphone. Its purpose is to run the higher-level perception and to provide an interface that allows the user to see the bounding boxes on the detected objects; which each bounding box will illustrated precisely on the detected objects and will has a text above it that tells the user about the class of the object and the confidence score of the predicted class. Also, the application interface shows the preprocessing time and the inference time of the model. The second component is a program that runs on the Arduino. It takes care of the low-level actuation. The Flutter application and the Arduino communicate via a serial communication protocol, which we used SoftwareSerial library [62] on Arduino and flutter_bluetooth_serial [63] library on flutter app . We took advantage of the computational power of the smartphone to process the input from the camera sensor of the mobile. We relied on the Tensorflow Lite infrastructure, which integrates seamlessly with smartphones and allowed us to deploy the deep learning model easily.

We used an Arduino Nano microcontroller to act as a bridge between the weeding robot body and the smartphone. Its main task is to handle the low-level control of the motors and laser light and to take appropriate action based on the received center point location from the mobile application and to make decisions to handle the readings from low-level robot input.

4.4.2 Location Mapping

The deep learning model classifies and localizes the objects. We wrote the logic of the application to send the location of the center point of the object if it is classified as a weed.

In Flutter code, we used the tflite model output, which is a list of objects in the following dictionary format:

```
          {
                 detectedClass: "weed",
                 confidenceInClass: 0.652,
                 rect: {
                   x: 0.15,
                   y: 0.33,
                   w: 0.80,
                   h: 0.27
            }
        }
```

Where x (left), y (top) indicate the coordinates of the top-left corner of the bounding box and w, h indicates the width and height of the bounding box. x, y, w, h are between (0, 1).

We created a Recognition class that transforms the location of the object detected to fit the camera screen. See Appendix C that shows the code. On each frame the application runs the function Tflite.detectObjectOnFrame() that detects all the weed objects in each frame and after that a list of Recognition will be created to store the all transformed locations of objects. For each recognition in the Recognition list, we calculate the center point of each bounding box in the frame and send it to the Arduino. The following code shows how we calculate the center point of each bounding box:

```
for (Recognition result in results) {
 var _left = result.renderLocation.left;
 var _top = result.renderLocation.top;
 var _width = result.renderLocation.width;
 var _height = result.renderLocation.height;
 var _x = _left + _width/2;
```

```
        var _y = _top + _height/2;
        int x = _x.floor();
        int y = _y.floor();
        _sendMessage("$x,$y!");
      }
```

The following figure clarifies how the object location rendered on the screen:

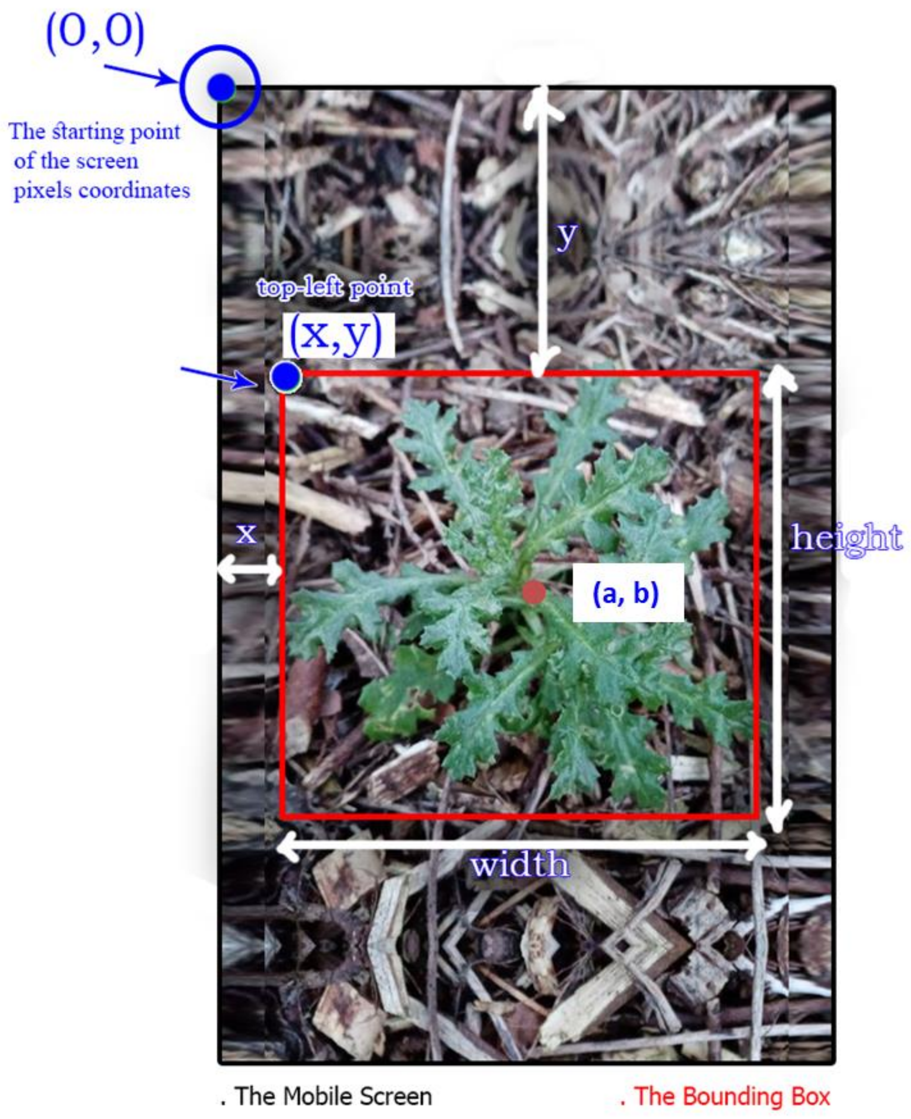


Figure 30: Object Bounding Box Location on Smartphone Screen

As shown in figure 30, we can calculate the center point coordinates (a, b) as follow:

$$a = \frac{x + width}{2}$$

Equation 2: Calculation the x-axis of Center Point

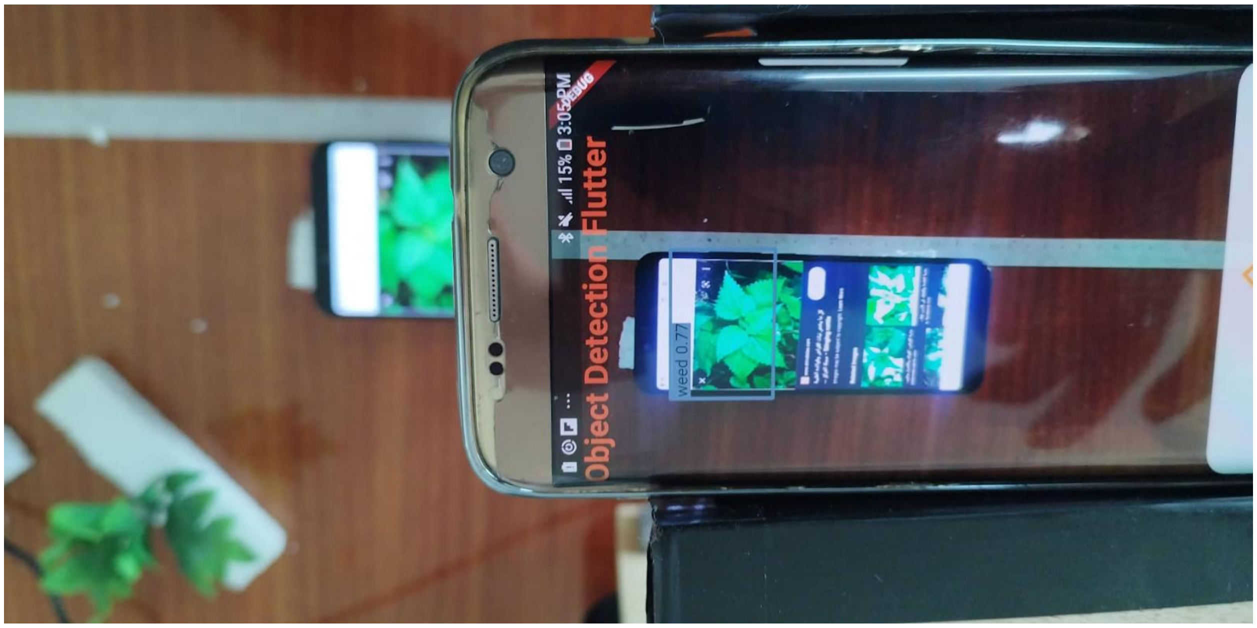
$$b = \frac{y + height}{2}$$

Equation 3: Calculation the y-axis of Center Point

After the mobile application sends the center point coordinates to the Arduino, the Arduino reads the coming characters and then stores each point coordinates as a string, then splits x coordinate and y coordinate and stores them in different string variables. After that each coordinate is converted to integer.

The received point is in pixels but we need to convert them in the actual dimensions that correspond to the dimensions in the image!

In order to do the mapping, we installed the smartphone on a fixed place on the top of the robot, since we need to keep the camera on a fixed space because if we change the camera place all the coordinates of the camera readings will change in respect to the laser position and other actuators. After that we put two weeds on the ground and then recorded the two center points coordinates from the camera (in pixels) as we recorded the real environment coordinates of the center points in centimeters. We have found the ratio of the pixels to centimeters in both dimensions.



Calculations:

We have to center points coordinates in pixels (position in the image) and centimeters (position in the real environment).

The first point has theses coordinates in pixels:

($X1_{pixels}$, $Y1_{pixels}$), and in centimeters: ($X1_{cm}$, $Y1_{cm}$)

The second point has theses coordinates in pixels:

($X2_{pixels}$, $Y2_{pixels}$), and in centimeters: ($X2_{cm}$, $Y2_{cm}$)

We found the ratio of the difference between the two points in pixel/cm for each dimension:

$$Y_{const} = \frac{Y2_{px} - Y1_{px}}{Y2_{cm} - Y1_{cm}}$$

$$X_{const} = \frac{X2_{px} - X1_{px}}{X2_{cm} - X1_{cm}}$$

Equation 4: The pixels to cms ratio constants in X and Y dimensions

We supposed an origin point in each dimension in the real environment coordinates. For the y-axis, the laser position is fixed relative to the position of the mobile camera, so we supposed that the laser light position on the ground in the y-axis as the origin point.

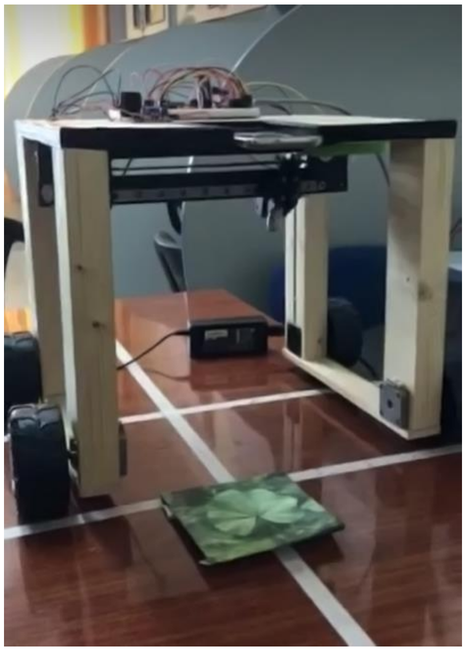


Figure 31: the position of the laser to the mobile camera

After we found the constant of the y-axis (Y_{const}) that represents the ratio of the pixels to cms, we can now implement the motion in y-axis by determining how many centimeters the robot needs to move in the y-dimension to reach the weed position. The following figure describes the position of the laser to the camera view on the mobile screen:

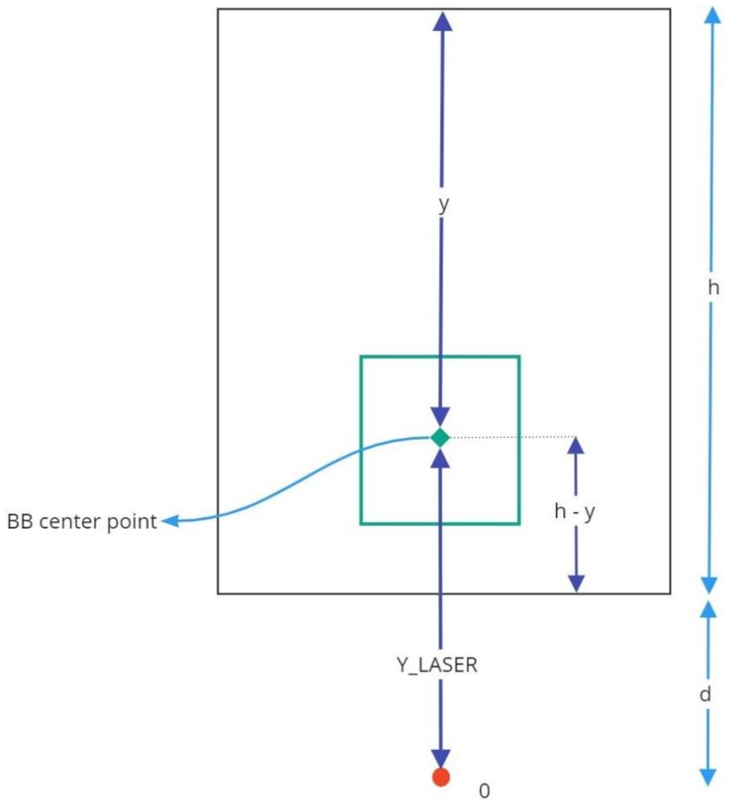


Figure 32: Calculating the number of centimeters to move in y-dimension based on the y coordinate of the BB center point

Where **h** represents the height of the mobile screen in pixels, and **y** represents the y coordinate of the bb center point position in pixels. The red point represents the laser light on the ground, and we need to calculate **Y_LASER** in centimeters to tell the robot how much the displacement will be to reach the weed position in the y axis where the laser will be exactly on the center point in y-dimension. **d** represents the distance between the laser light and the most down pixel the camera can see in centimeters.

The following equation shows how we calculated the **Y_LASER**:

$$Y_{laser} = (h - y) \cdot Y_{const} + d$$

Equation 5: required number of centimeters in y-axis

After we finished from the y-axis, we moved to implement the motion in the x-axis. We had already calculated the constant of the x-axis (X_{const}), so it was also easy to calculate how many centimeters the laser on the linear-slide needs to move to reach the position of the weed.

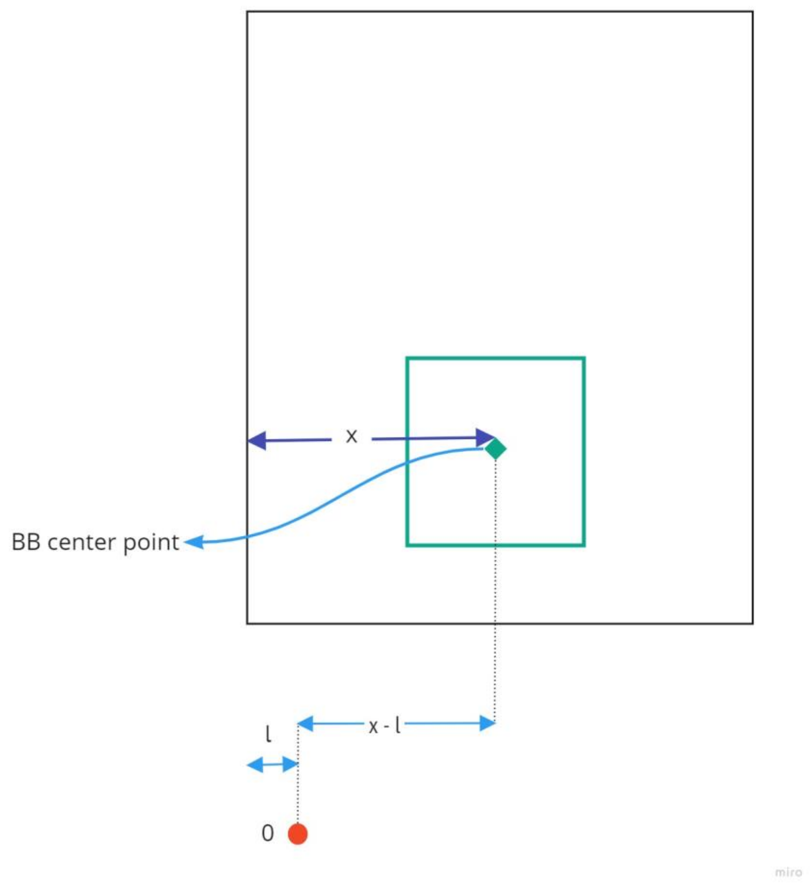


Figure 33: Calculating the number of centimeters to move in x-dimension based on the x coordinate of the BB center point

Where **x** represents the x coordinate of the bb center point position in pixels, **l** represents the distance between the laser light and the most left pixel the camera can see in centimeters. **X_LASER** is the amount of centimeters to reach the goal position of the weed object.

So, to calculate how many centimeters the laser needs to move in the x-axis to reach the weed position by the following equation:

$$X_{laser} = (x \cdot X_{const}) - l$$

Equation 6: required number of centimeters in x-axis

Chapter 5 Testing and Results

5.1 Overview

This chapter discusses the testing process of the main system components and presents the results obtained from the implementation of the system.

5.2 Hardware Testing

This section details the testing process of the main hardware components and how we overcame the problems we faced in the Implementation of the system hardware.

5.2.1 Testing Stepper Motors with Motor Drivers

In testing stepper motors, we first used an ohmmeter to indicate if a winding is burnt up and what type of stepper motor we have (bipolar or unipolar), were each type of stepper motor controlled by a special driver circuit. The two bipolar stepper motors that we connected to the wheels have four leads, and the Unipolar stepper motor of the arm has six leads. Using the ohmmeter, we checked the resistance of the windings, to find the wire pair of each winding. On a bipolar the resistance for both windings should be the same in both directions. In a unipolar winding, the resistance from each phase to com should be the same in both directions.

After we checked the motor with the ohmmeter, we used two 5 volt / 1 Ampere power banks and connected them in series to get a 10 Volts power to step the motor through its paces. When we started testing the unipolar stepper motor of the arm it worked well, but when we tested the bipolar stepper motor it was vibrating without moving forward or backward. Because of this, we supposed that the motor did not get enough current, so we used the power supply and we increased the current to 1.8 Amp and then the motor worked and the shaft started rotating, so we concluded that the problem was in the current.

When we tried to let the robot move we noticed that the robot stayed stopped and did not move. It was because of the torque of the wheels motors; since the robot weight was about 5 Kilograms and this required more torque to hold the weight. We try to increase the torque of the motors by decreasing the speed through reducing the number of revolutions per minute (rpm); since we notice that when we reduce the rpm the motors consume more current and produce more torque, but by doing this the robot still stops on some points.

We are was trying to find a way to increase the stepper motors consumption of the bipolar stepper motors without adding another two stepper motors, and we have found that there is a potentiometer on the EasyDriver chip that control the maximum amount of current that gives to the stepper motor, and when we adjust the potentiometer to the maximum current we get a sufficient torque of the motors that can hold the robot weight.

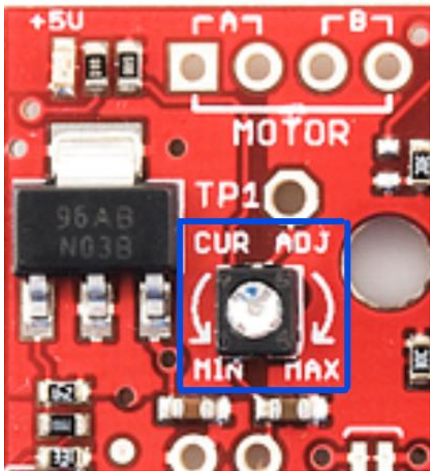


Figure 34: The potentiometer on EasyDriver chip

5.2.2 Testing Arduino Nano Microcontroller

The Arduino Nano microcontroller was the first component that we tested in our project, since it will control all other electronics components and will communicate with the smartphone via the Bluetooth module, so it is very important to ensure that all pins are working well and there is no errors on any GPIO pins, and also to verify that we can upload the sketch on the microcontroller chip without any faults and ensuring the functionality of the chip.

We started by uploading a sketch that controlled two leds and connected two leds to the Arduino pins to check the pins and the board functionality. We used leds to check the different analog and digital pins are working well, as we tried writing different values at different digital and analog pins and observing the output using a Multimeter.

We also tested the serial function of the Arduino by using a simple application that can chat with the Arduino through the serial communication port.

To ensure that the Arduino can handle the different processing tasks at the same time without any problem, we have tested the Arduino with the different components of the system simultaneously.

5.2.3 Testing HC-06 Bluetooth Module

One HC-06 Bluetooth module has connected to the Arduino Nano chip. We tested the communication between the mobile application and the Arduino Nano via the HC-06 by sending messages from the mobile application to Arduino and it succeeded.

We tried to use the SoftwareSerial library on the Arduino to choose any two digital pins to connect with the HC-06 to create the serial communication between the Arduino Nano and the smartphone and it worked for our purpose.

5.3 Software Testing and Result

In our project, the software testing goes through different stages, and sometimes the testing stages are overlapped. These are the three stages of software testing in the weeding robot system prototype: testing the software of the Arduino and the different libraries that can help up in the low-level components controlling, testing the deep learning model (accuracy, precision, inference speed, and the ability to deploy on a smartphone with limited resources), and the last stage is testing the mobile application and the different parts in the application.

In the following subsections we will give more details about each stage in the software testing:

5.3.1 Arduino Software Testing

A simple methodology used to test the software on the Arduino: we first upload the code of each component separately and ensure that all functionalities of the software are working appropriately, then we do integration between different components and ensure that the software works in a way as intended and as expected. When we were found a problem in a point in the operation sequence or in the logic of the program we were trying to write code in different way,(i.e. use interrupts in the code for some functions) and sometimes we were tried using different libraries(i.e. use Stepper library instead of AccelStepper to control the stepper motors in appropriate procedure).

We tested the serial communication and trying to read a coordinate point as a String, but the Serial.read() function reads only one byte at a time (one character) so we trying to send an end character with each point from the mobile application and store each point as a String variable, so we use an iteration method to read characters using Serial.read() function and store all the characters in a string variable and when the condition of read end line character is achieved then it will move to store new point coordinates string variable.

5.3.2 Model Testing and Result

5.3.2.1 Inference Time

Inference time is the amount of time it takes the model to process test data and makes a prediction at run time and it’s measured in ms. Most real-time applications require fast inference time, which means lower inference time is better.

As we mentioned in section 4.3.2.1, YoloV4 and YoloV4Tiny were successfully deployed on the mobile application, so we measured the inference time for each one to choose the faster one.

YoloV4 has an average of 5000ms where the YoloV4Tiny has an average of 400ms, the inference time of these models varied greatly, so the choice was for the lower value which is the YoloV4Tiny.

5.3.2.2 Mean Average Precision (mAP)

The mean average precision (mAP) is a metric used to measure the accuracy of object detectors over all classes in a specific database. The mAP is simply the average AP over all classes [64], that is:

$$mAP = \frac{1}{N}\sum_{i=1}^{N} AP_i$$

Equation 7: Mean Average Precision

In order to find the best performing model for the task of detecting, the efficientdet_lite, YOLOv4, YOLOv4-tiny, and YOLOv5 ML models were trained and evaluated in terms of mAP performance. The performance of the trained models is displayed in Table 6, where all models have been trained on the training data and tested on the test data.

Model	mAP
efficientdet_lite	54.06%
YOLOv4	75.35%
YOLOv4-tiny	50.67%
YOLOv5	78.3%

Table 6: The mAP performance of the trained models.

Models' results vary from one to another in the accuracy, inference speed, size, and training time. First we started with the efficientdet_lite model that is a light model supported by the tflite to deploy on mobile; in our prototype we used a smartphone to deploy the model. The result of the efficientdet_lite model was not good enough, so we thought about trying a full version model to test if the problem is in the data or in the lite model. We tried YOLOv5: as we previously talked, YOLO is one of the fastest object detection methods with good real-time performance and high accuracy. YoloV4 and YoloV5 give better performance than the efficientdet_lite model.

But in the mobile development, the YoloV5 was not working due to the version of the dependencies, instead we trained the tiny version of YoloV4, the mAP seems not efficient but in the mobile environment, this mean-average precision is appropriate.

5.3.3 Mobile Application

We tested the components of the software which we describe in section 4.3.3.5. The methodology of the testing was simple, we ran each page separately and checked it, then we integrated the software module, we started by integrating the Bluetooth module and the camera module. Finally we ran the whole project and ensured that everything was working correctly.

5.4 Hardware Implementation Results

By the end of the implementation process, the prototype of the weeding robot was done and worked successfully. The following figure shows the result of the hardware prototype:

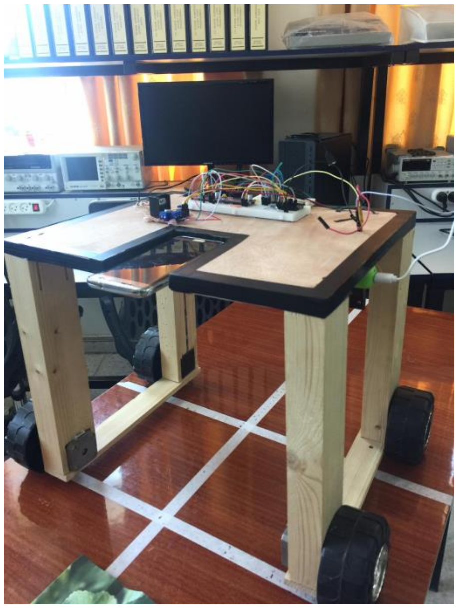
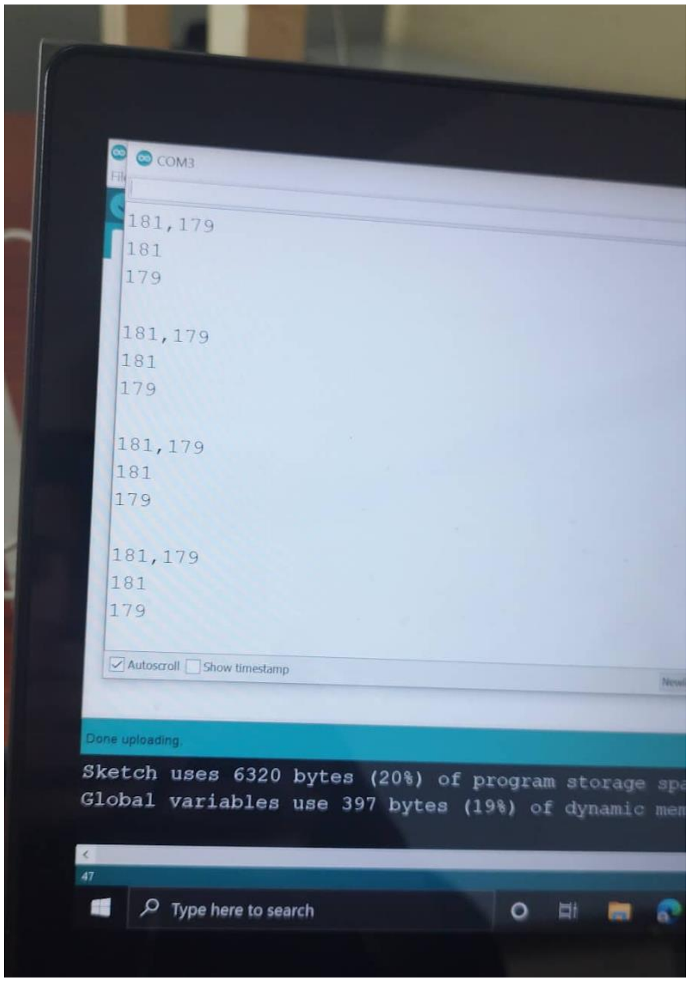


Figure 35: Hardware Implementation Result

We took the coordinates of the detected objects in each frame from the tflite model output, and sent them successfully to the microcontroller. The following figure shows the serial monitor of the Arduino when receiving coordinates from the mobile application:



```
181,179
181
179

181,179
181
179

181,179
181
179

181,179
181
179
```

```
Sketch uses 6320 bytes (20%) of program storage spa
Global variables use 397 bytes (19%) of dynamic mem
```

Figure 36: Data Received from the smartphone and separated in x & y coordinates

After receiving the location data, the robot was able to reach the weed location accurately and activate the laser on the weed for a specific time. The process of detecting the object and move to its position is shown in the following figures:

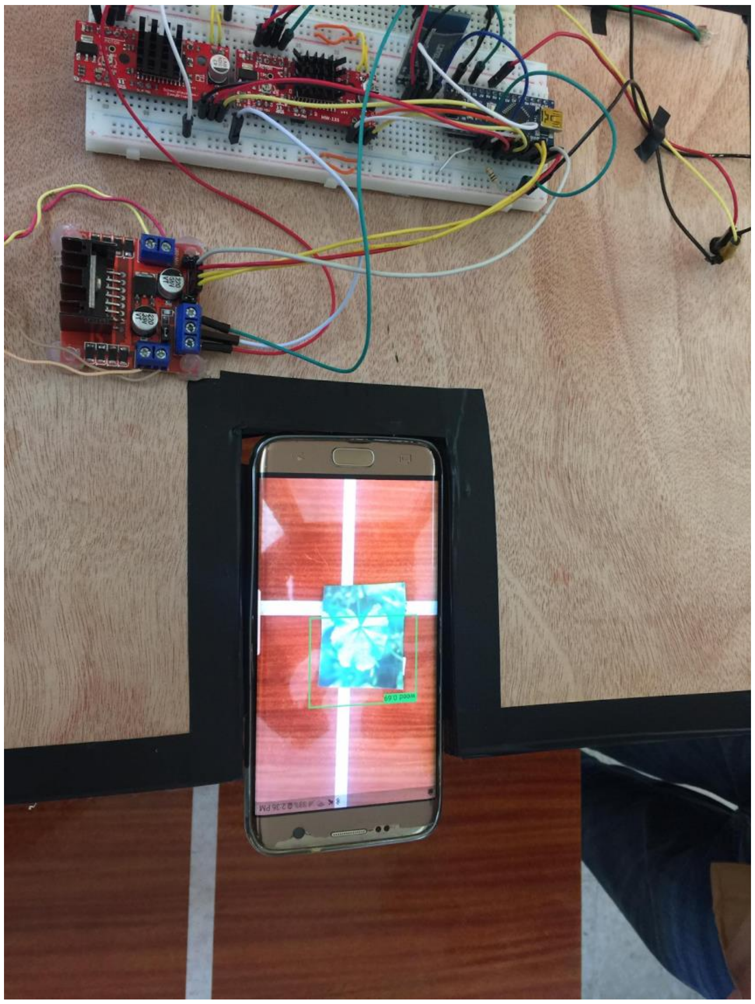


Figure 37: Detecting the object on the smartphone

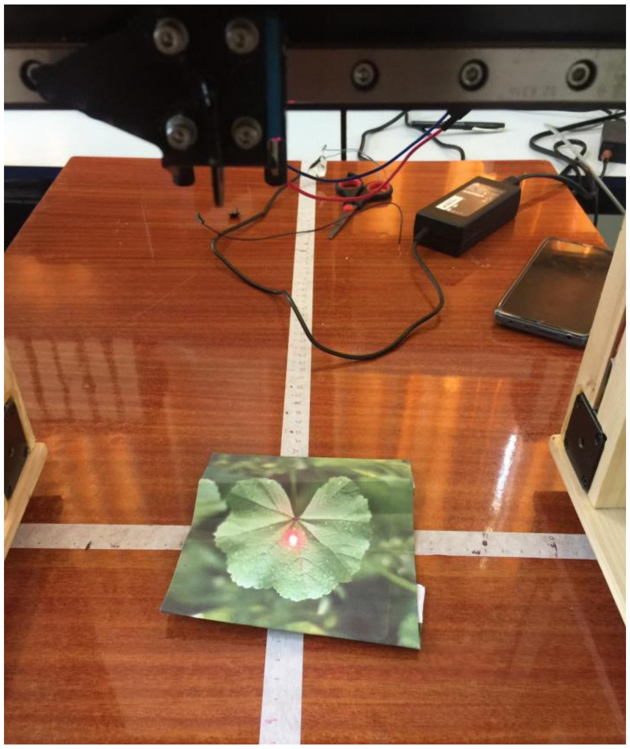


Figure 38: Move to the weed position and activate the laser

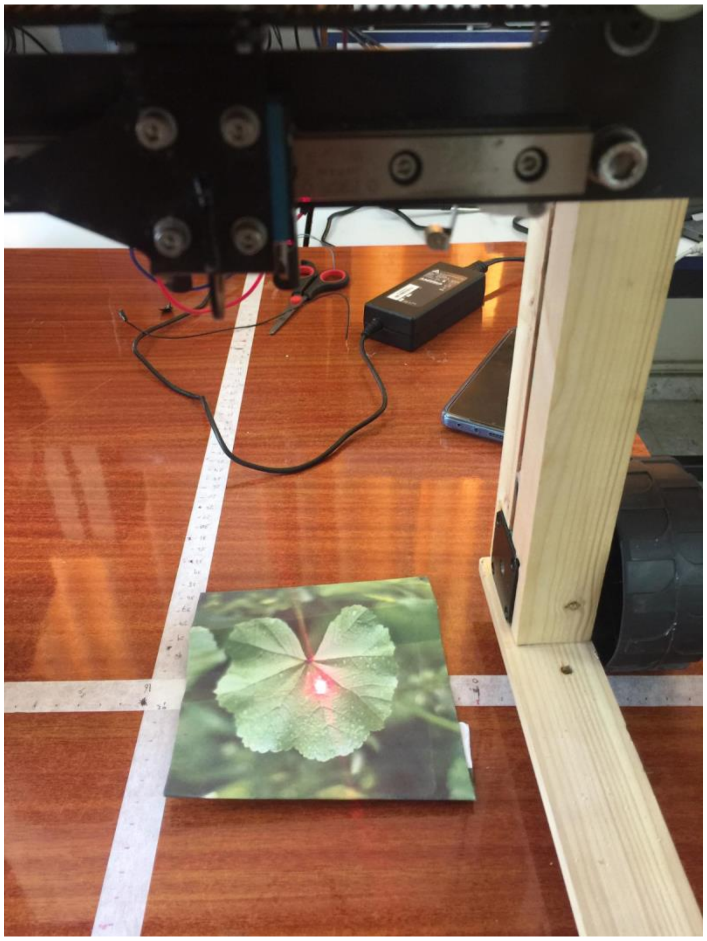


Figure 39: Detect and move to the weed in different location

Conclusion

In this project we have presented an approach to building a prototype of a weeding robot using deep learning and based on the Arduino microcontroller with a Bluetooth module and smartphone communication.

Using multiple base models and object detection frameworks, we successfully trained and implemented a variety of models that could be used for real-time detection of weeds in a Flutter application. Using Transfer Learning, the network could be trained on a relatively small amount of data while still achieving good mAP scores.

We created a communication between the smartphone and the Arduino using UART serial communication protocol to send the objects coordinates from the mobile application to the microcontroller.

The developed weed detection system was presented and tested on real-world data and good confidence scores on detection were achieved. It can be concluded that higher values of mAP could be achieved with more steps with the right hyperparameters.

Future Work

Future work in this project could include investigations on choosing a method for optimization the system, so here some improvements we can add to our project:

- Using a microcomputer that has high processing power which enables us to deploy a large size model that has a high accuracy to get high resolution of detection with a high processing speed.
- Autonomous navigation.
- Using 3D arm to control the laser position in more efficient way
- Retrain the model with more images of new weed types and different crops.
- Train the model on insect data and add another arm that carries insecticide to kill the insects.

References

[1] Britannica, The Editors of Encyclopedia. "Weed". Encyclopedia Britannica, 19 Nov. 2020.

[2] B. B. Park et al., “Analysis of the Current Status of Weeding Operation and Crop Tree Growth Across Planting Periods,” Journal of Korean Society of Forest Science, vol. 109, no. 2, pp. 179–188, Jun. 2020.

[3] Ane, T., & Yasmin, S. (2019). Agriculture in the Fourth Industrial Revolution. Annals of Bangladesh Agriculture, 23(2), 115–122.

[4] Liakos, K.; Busato, P.; Moshou, D.; Pearson, S.; Bochtis, D. Machine Learning in Agriculture: A Review. Sensors 2018, 18, 2674.

[5] Liebman,M.; Baraibar, B.; Buckley, Y.; Childs, D.; Christensen, S.; Cousens, R.; Eizenberg, H.; Heijting, S.; Loddo, D.;Merotto, A.; et al. Ecologically sustainable weed management: How do we get from proof-of-concept to adoption? Ecol. Appl. 2016, 26, 1352–1369.

[6] Huang MH, Rust RT. Artificial Intelligence in Service. Journal of Service Research. 2018;21(2):155-172. doi:10.1177/1094670517752459

[7] Vrontis D, Christofi M, Pereira V, Tarba S, Makrides A, Trichina E. Artificial intelligence, robotics, advanced technologies and human resource management: a systematic review. The International Journal of Human Resource Management. 2021;33(6):1-30. doi:10.1080/09585192.2020.1871398

[8] Murphy, K.P. Machine Learning: A Probabilistic Perspective; MIT Press: Cambridge, MA, USA, 2012; ISBN 978-0-262-01802-9.

[9] Willi M, Pitman RT, Cardoso AW, et al. Identifying animal species in camera trap images using deep learning and citizen science. Gaggiotti O, ed. Methods in Ecology and Evolution. 2018;10(1):80-91. doi:10.1111/2041-210x.13099

[10] Brownlee J. What is the Difference Between Test and Validation Datasets? Machine Learning Mastery. July 26, 2017. https://machinelearningmastery.com/difference-test-validation-datasets/

[11] Simon, H. Neural Networks: A Comprehensive Foundation; McMaster University: Hamilton, ON, Canada, 2005; p. 823. 72. Park, S.H. Artificial Intelligence in Medicine: Beginner’s Guide. J. Korena Soc Radiol. 2018, 78, 301–308.

[12] Roberts D, Yaida S, Hanin B. The Principles of Deep Learning Theory.; 2021. Accessed April 27, 2022. https://arxiv.org/pdf/2106.10165.pdf

[13] Kavlakoglu E. AI vs. Machine Learning vs. Deep Learning vs. Neural Networks: What’s the Difference? IBM. Published May 27, 2020. https://www.ibm.com/cloud/blog/ai-vs-machine-learning-vs-deep-learning-vs-neural-networks

[14] Yiu T. Understanding Neural Networks. Medium. Published June 2, 2019. https://towardsdatascience.com/understanding-neural-networks-19020b758230

[15] Shah TM, Nasika DPB, Otterpohl R. Plant and Weed Identifier Robot as an Agroecological Tool Using Artificial Neural Networks for Image Identification. Agriculture. 2021;11(3):222. doi:10.3390/agriculture11030222

[16] Wu Z, Chen Y, Zhao B, Kang X, Ding Y. Review of Weed Detection Methods Based on Computer Vision. Sensors. 2021; 21(11):3647. https://doi.org/10.3390/s21113647

[17] Olivas, E.S.; Guerrero, J.D.M.; Martinez-Sober, M.; Magdalena-Benedito, J.R.; Serrano, L. Handbook of Research on machine Learning Applications and Trends: Algorithms, Methods, and Techniques; IGI Global: Hershey, PA, USA, 2009; ISBN 1605667676.

[18] Kaya, A.; Keceli, A.S.; Catal, C.; Yalic, H.Y.; Temucin, H.; Tekinerdogan, B. Analysis of transfer learning for deep neural network based plant classification models. Comput. Electron. Agric. 2019, 158, 20–29.

[19] Pan, S.J.; Yang, Q. A survey on transfer learning. IEEE Trans. Knowl. Data Eng. 2009, 22, 1345–1359.

[20] Zhao, W. Research on the deep learning of small sample data based on transfer learning. AIP Conf. Proc. 2017, 1864, 20018.

[21]Jason Brownlee. A Gentle Introduction to Object Recognition With Deep Learning. Machine Learning Mastery. Published July 5, 2019. https://machinelearningmastery.com/object-recognition-with-deep-learning/

[22] Fang W, Wang L, Ren P. Tinier-YOLO: A Real-time Object Detection Method for Constrained Environments. IEEE Access. Published online 2019:1-1. doi:10.1109/access.2019.2961959

[23] Alsing O. Mobile Object Detection using TensorFlow Lite and Transfer Learning. 2018. https://www.diva-portal.org/smash/get/diva2:1242627/FULLTEXT01.pdf

[24] Kusnetsoff D. Mobile Real-Time Object Detection with Flutter.2021.https://www.theseus.fi/bitstream/handle/10024/499717/Kusnetsoff_Daniel.pdf?sequence=2

[25] Jiang P, Ergu D, Liu F, Cai Y, Ma B. A Review of Yolo Algorithm Developments. Procedia Computer Science. 2022;199:1066-1073. doi:10.1016/j.procs.2022.01.135

[26] Geethapriya S, N Duraimurugan, S.P. Chokkalingam. Real-Time Object Detection with Yolo. International Journal of Engineering and Advanced Technology (IJEAT). February 2019. https://www.ijeat.org/wp-content/uploads/papers/v8i3S/C11240283S19.pdf

[27] Liu W, Anguelov D, Erhan D, et al. SSD: Single Shot MultiBox Detector. Computer Vision – ECCV 2016. Published online 2016:21-37. doi:10.1007/978-3-319-46448-0_2

[28] Kanimozhi S, Gayathri G, Mala T. Multiple Real-time object identification using Single shot Multi-Box detection. IEEE Xplore. doi:10.1109/ICCIDS.2019.8862041

[29] Younis A, Shixin L, Jn S, Hai Z. Real-Time Object Detection Using Pre-Trained Deep Learning Models MobileNet-SSD. Proceedings of 2020 the 6th International Conference on Computing and Data Engineering. Published online January 4, 2020. doi:10.1145/3379247.3379264

[30] L.M. van Kollenburg-Crisan, J. Bontsema, P. Wennekes, Mechatronic System for Automatic Harvesting of Cucumbers, IFAC Proceedings Volumes, Volume 31, Issue 12, 1998; ISSN 1474-6670.

[31] Xiong Y, Ge Y, Liang Y, Blackmore S. Development of a prototype robot and fast path-planning algorithm for static laser weeding. Computers and Electronics in Agriculture. 2017;142:494-503. doi:10.1016/j.compag.2017.11.023

[32] Perez-Ruiz, M., Brenes, R., Urbano, J.M. et al. Agricultural residues are efficient abrasive tools for weed control. Agron. Sustain. Dev. 38, 18 (2018). https://doi.org/10.1007/s13593-018-0494-6

[33] 1. Peruzzi A, Martelloni L, Frasconi C, Fontanelli M, Pirchio M, Raffaelli M. Machines for non-chemical intra-row weed control in narrow and wide-row crops: a review. Journal of Agricultural Engineering. 2017;48(2):57. doi:10.4081/jae.2017.583

[34] Brownlee, J. (2017). Deep Learning with Python: Develop Deep Learning Models on Theano and TensorFlow Using Keras. Machine Learning Mastery https://books.google.ps/books?id=K-ipDwAAQBAJ&lpg=PP1&hl=ar&pg=PP1#v=onepage&q&f=false (Original work published 2016)

[35] TensorFlow Lite guide. TensorFlow. https://www.tensorflow.org/lite/guidet

[36] https://pytorch.org/

[37] Ketkar N. (2017) Introduction to PyTorch. In: Deep Learning with Python. Apress, Berkeley, CA. https://doi.org/10.1007/978-1-4842-2766-4_12

[38] Howse, J. (2013). OpenCV computer vision with python: Learn to capture videos, manipulate images, and track objects with python using the opencv library. Packt Publishing, Limited.

[39] flutter documentation: https://flutter.dev/

[40] https://dart.dev/overview

[41] M Florance Mary and D Yogaraman 2021 J. Phys.: Conf. Ser. 1979 012027

[42] Reiser D, Sehsah E-S, Bumann O, Morhard J, Griepentrog HW. Development of an Autonomous Electric Robot Implement for Intra-Row Weeding in Vineyards. Agriculture. 2019; 9(1):18. https://doi.org/10.3390/agriculture9010018

[43] Shah TM, Nasika DPB, Otterpohl R. Plant and Weed Identifier Robot as an Agroecological Tool Using Artificial Neural Networks for Image Identification. *Agriculture*. 2021;11(3):222. doi:10.3390/agriculture11030222.

[44] Kargar AHB, Shirzadifar AM. Automatic weed detection system and smart herbicide sprayer robot for corn fields. 2013 First RSI/ISM International Conference on Robotics and Mechatronics (ICRoM). Published online February 2013. doi:10.1109/icrom.2013.6510152

[45] Kanagasingham, S., Ekpanyapong, M. & Chaihan, R. Integrating machine vision-based row guidance with GPS and compass-based routing to achieve autonomous navigation for a rice field weeding robot. Precision Agric 21, 831–855 (2020).

[46] Fennimore SA, Cutulle M. Robotic weeders can improve weed control options for specialty crops. Pest Management Science. Published online February 28, 2019. doi:10.1002/ps.5337.

[47] Wu X, Aravecchia S, Lottes P, Stachniss C, Pradalier C. Robotic weed control using automated weed and crop classification. Journal of Field Robotics. 2020;37(2):322-340. doi:10.1002/rob.21938

[48] Marx C, Barcikowski S, Hustedt M, Haferkamp H, Rath T. Design and application of a weed damage model for laser-based weed control. *Biosystems Engineering*. 2012;113(2):148-157. doi:10.1016/j.biosystemseng.2012.07.002

[49] https://www.farnell.com/datasheets/1682238.pdf

[50] https://docs.arduino.cc/learn/built-in-libraries/software-serial

[51] https://www.motioncontrolproducts.co.uk/productimages/cms/PDF/FL42STH_datasheet.pdf

[52] data sheet of SST42D2120 https://www.datasheetarchive.com/pdf/download.php?id=fe7cb1b5a9daa990e6683cdf55439492077971&type=P&term=SHINANO%2520KENSHI%2520STH

[53] https://www.sparkfun.com/datasheets/Robotics/L298_H_Bridge.pdf

[54] https://www.electronicoscaldas.com/datasheet/A3967-EDMOD_Manual.pdf

[55] [https://www.electronicoscaldas.com/datasheet/A3967-EDMOD_Manual.pdf

[56]https://www.olimex.com/Products/Components/RF/BLUETOOTH-SERIAL-HC-06/resources/hc06.pdf

[57] https://www.escarda.eu/technology/

[58] https://www.gbif.org/

[59] https://www.makesense.ai/

[60] techzizou007. Train a custom YOLOv4-tiny detector using Google Colab. TECHZIZOU. Published March 2, 2021. https://techzizou.com/train-a-custom-yolov4-tiny-object-detector-using-google-colab-tutorial-for-beginners/

[61] GitHub - theAIGuysCode/tensorflow-yolov4-tflite at morioh.com. GitHub. https://github.com/theAIGuysCode/tensorflow-yolov4-tflite?ref=morioh.com&utm_source=morioh.com

[62] https://docs.arduino.cc/learn/built-in-libraries/software-serial

[63] https://pub.dev/packages/flutter_bluetooth_serial

[64] R. Padilla, S. L. Netto and E. A. B. da Silva, "A Survey on Performance Metrics for Object-Detection Algorithms," 2020 International Conference on Systems, Signals and Image Processing (IWSSIP), 2020, pp. 237-242, doi: 10.1109/IWSSIP48289.2020.9145130.

[65] https://docs.arduino.cc/hardware/nano

Appendices

A. process.py script

```
import glob, os
# Current directory
current_dir = os.path.dirname(os.path.abspath(__file__))

print(current_dir)
current_dir = 'data/obj'
# Percentage of images to be used for the test set
percentage_test = 10;
# Create and/or truncate train.txt and test.txt
file_train = open('data/train.txt', 'w')
file_test = open('data/test.txt', 'w')
# Populate train.txt and test.txt
counter = 1
index_test = round(100 / percentage_test)
for pathAndFilename in glob.iglob(os.path.join(current_dir, "*.jpg")):
  title, ext = os.path.splitext(os.path.basename(pathAndFilename))
  if counter == index_test:
    counter = 1
    file_test.write("data/obj" + "/" + title + '.jpg' + "\n")
  else:
    file_train.write("data/obj" + "/" + title + '.jpg' + "\n")
    counter = counter + 1
```

B. Training Yolov4-tiny detector

```
# Clone Darknet git repository
!git clone https://github.com/AlexeyAB/darknet

# Mount drive and link your folder
%cd ..
from google.colab import drive
drive.mount('/content/gdrive')
!ln -s /content/gdrive/My Drive/ /mydrive

# Make changes in the makefile to enable OPENCV and GPU
%cd darknet/
!sed -i 's/OPENCV=0/OPENCV=1/' Makefile
!sed -i 's/GPU=0/GPU=1/' Makefile
!sed -i 's/CUDNN=0/CUDNN=1/' Makefile
!sed -i 's/CUDNN_HALF=0/CUDNN_HALF=1/' Makefile
!sed -i 's/LIBSO=0/LIBSO=1/' Makefile

# Run make command to build darknet
!make

# Copy all the files from the 'yolov4-tiny' folder to the 'darknet'
directory in Colab VM
```

```
!cp /mydrive/yolov4-tiny/obj.zip ../
!unzip ../obj.zip -d data/

!cp /mydrive/yolov4-tiny/yolov4-tiny-custom.cfg ./cfg
!cp /mydrive/yolov4-tiny/obj.names ./data
!cp /mydrive/yolov4-tiny/obj.data  ./data
!cp /mydrive/yolov4-tiny/process.py ./

# Run the process.py python script to create the train.txt & test.txt
files inside the data folder
!python process.py

# Download the pre-trained YOLOv4-tiny weights
!wget
https://github.com/AlexeyAB/darknet/releases/download/darknet_yolo_v4_pre/
yolov4-tiny.conv.29

# Training
!./darknet detector train data/obj.data cfg/yolov4-tiny-custom.cfg yolov4-
tiny.conv.29 -dont_show -map

# Define helper function imShow
def imShow(path):
import cv2
import matplotlib.pyplot as plt
%matplotlib inline
image = cv2.imread(path)
height, width = image.shape[:2]
resized_image = cv2.resize(image,(3*width, 3*height), interpolation =
cv2.INTER_CUBIC)
fig = plt.gcf()
fig.set_size_inches(18, 10)
plt.axis("off")
plt.imshow(cv2.cvtColor(resized_image, cv2.COLOR_BGR2RGB))
plt.show()

# Test your custom Object Detector
%cd cfg
!sed -i 's/batch=64/batch=1/' yolov4-tiny-custom.cfg
!sed -i 's/subdivisions=16/subdivisions=1/' yolov4-tiny-custom.cfg
%cd ..

# Check mAP (mean average precision)
!./darknet detector map data/obj.data cfg/yolov4-tiny-custom.cfg
/mydrive/yolov4-tiny/training/yolov4-tiny-custom_best.weights -points 0

# Run detector on an image
!./darknet detector test data/obj.data cfg/yolov4-tiny-custom.cfg
/mydrive/yolov4-tiny/training/yolov4-tiny-custom_best.weights
/mydrive/mask_test_images/image1.jpg -thresh 0.3
imShow('predictions.jpg')
```

C. recognition.dart

```
import 'dart:math';

import 'package:flutter/cupertino.dart';

import '/ui/camera_view_singleton.dart';


/// Represents the recognition output from the model

class Recognition implements Comparable<Recognition> {

 /// Index of the result

 int _id;

 /// Label of the result

 String _label;


 /// Confidence [0.0, 1.0]

 double _score;

 /// Location of bounding box rect

 /// The rectangle corresponds to the raw input image

 /// passed for inference

 Rect _location;

 Recognition(this._id, this._label, this._score, [this._location]);

 int get id => _id;

 String get label => _label;

 double get score => _score;

 Rect get location => _location;

 /// Returns bounding box rectangle corresponding to the

 /// displayed image on screen

 /// This is the actual location where rectangle is rendered on

 /// the screen

 Rect get renderLocation {

   // ratioX = screenWidth / imageInputWidth

   // ratioY = ratioX if image fits screenWidth with aspectRatio = constant


   double ratioX = CameraViewSingleton.ratio;

   double ratioY = ratioX;


   double transLeft = max(0.1, location.left * ratioX);

   double transTop = max(0.1, location.top * ratioY);
```

```
    double transWidth = min(
        location.width * ratioX, CameraViewSingleton.actualPreviewSize.width);
    double transHeight = min(
        location.height * ratioY,
CameraViewSingleton.actualPreviewSize.height);

    Rect transformedRect =
    Rect.fromLTWH(transLeft, transTop, transWidth, transHeight);
    return transformedRect;
  }

  @override
  String toString() {
    return 'Recognition(id: $id, label: $label, score: ${(score *
100).toStringAsPrecision(3)}, location: $location)';
  }

  @override
  int compareTo(Recognition other) {
    if (this.score == other.score) {
      return 0;
    } else if (this.score > other.score) {
      return -1;
    } else {
      return 1;
    }
  }
}
```