

**Project Stage - II (CS802PC)**

On

**PEST DETECTION AND CONTROL SYSTEM USING DEEP  
LEARNING AND IOT**

Submitted in partial fulfilment of the  
requirements for the award of the degree of

**Bachelor of Technology**

in

**Computer Science and Engineering**

by

**Mr. Lalith Chakradhar R. (18261A05F5)**

**Mr. P. Shiva Rama Krishna (18261A05G3)**

Under the Guidance of

**Ms. K. Shirisha** (Assistant Professor)

**Ms. Vijaylaxmi C Handaragall** (Assistant Professor)

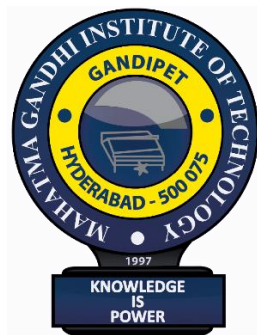


**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING  
MAHATMA GANDHI INSTITUTE OF TECHNOLOGY  
(Affiliated to Jawaharlal Nehru Technological University Hyderabad)  
Gandipet, Hyderabad-500075, Telangana (India)  
2021-2022**

# MAHATMA GANDHI INSTITUTE OF TECHNOLOGY

(Affiliated to Jawaharlal Nehru Technological University  
Hyderabad) Gandipet, Hyderabad – 500075, Telangana.

## CERTIFICATE



This is to certify that Project Stage - II (CS802PC) entitled “**PEST DETECTION AND CONTROL SYSTEM USING DEEP LEARNING AND IOT**” is being submitted by **Lalith Chakradhar R.** bearing **Roll No: 18261A05F5** and **P. Shiva Rama Krishna** bearing **Roll No:18261A05G3** in partial fulfilment for the award of **B.Tech in Computer Science and Engineering** to **Jawaharlal Nehru Technological University Hyderabad** is a record of bonafide work carried out under the supervision of **Ms. K. Shirisha, Assistant Professor, Department of CSE** and **Ms. Vijaylaxmi C Handaragall, Assistant Professor, Department of CSE.**

The results embodied in this project have not been submitted to any other University or Institute for the award of any degree or diploma.

Project Guide	Project Guide	Project Coordinator	Head of Department
<b>Ms.K. Shirisha</b>	<b>Ms.Vijaylaxmi C Handaragall</b>	<b>Dr.M. Rama Bhai</b>	<b>Dr.C.R.K. Reddy</b>
Assistant Professor	Assistant Professor	Professor, Dept. of CSE	Professor, Dept. of CSE

**External Examiner**

## **DECLARATION**

This is to certify that the work reported in this project titled “**PEST DETECTION AND CONTROL SYSTEM USING DEEP LEARNING AND IOT**” is a record of work done by **Mr. Lalith Chakradhar R.** and **Mr. P. Shiva Rama Krishna** in the Department of Computer Science and Engineering, Mahatma Gandhi Institute of Technology, Hyderabad.

No part of the work is copied from books/journals/internet and wherever the portion is taken, the same has been duly referred in the text. The report is based on the work done entirely by us and not copied from any other source.

**Lalith Chakradhar R. (18261A05F5),  
P. Shiva Rama Krishna (18261A05G3)**

## ACKNOWLEDGEMENT

We would like to express our sincere thanks to **Dr. K. Jaya Sankar, Principal MGIT**, for providing the working facilities in the college.

We express our sincere thanks and gratitude to **Dr. C.R.K. Reddy, Professor & Head of Department**, Department of Computer Science and Engineering, for all the timely support and valuable suggestions during the period of project.

We are extremely thankful to Project Coordinators, **Dr. M. Rama Bai, Professor, Dept. of CSE, Dr. M. Sreevani, Associate Professor, Dept. of CSE and Dr. Mahesh Kumar, Associate Professor, Dept. of CSE** for their encouragement and support throughout the project.

We are extremely thankful and indebted to Project Guides, **Ms. K. Shirisha, Assistant Professor, Dept. of CSE and Ms. Vijaylaxmi C Handaragall, Assistant Professor, Dept. of CSE** for their constant guidance, encouragement and moral support throughout the project.

Finally, we would also like to thank all the faculty and staff of CSE Department who helped us directly or indirectly, for completing this project.

**Lalith Chakradhar R. (18261A05F5),  
P. Shiva Rama Krishna (18261A05G3)**

# TABLE OF CONTENTS

Certificate	i
Declaration	ii
Acknowledgement	iii
List of Tables	vi
List of Figures	vi
ABSTRACT	ix
<b>1. INTRODUCTION</b>	<b>10</b>
1.1 Problem Definition	11
1.2 Existing System	11
1.3 Proposed System	11
1.4 System Requirements	12
1.4.1 Software Requirements	12
1.4.2 Hardware Requirements	12
<b>2. LITERATURE SURVEY ON PEST DETECTION AND CONTROL</b>	
<b>SYSTEM USING DEEP LEARNING AND IoT</b>	<b>13</b>
<b>3. DESIGN AND METHODOLOGY OF PEST DETECTION AND</b>	
<b>CONTROL SYSTEM USING DEEP LEARNING AND IoT</b>	<b>17</b>
3.1.1 System Architecture	17
3.1.2 Pesticide Sprayer Bot Architecture	18
3.2 UML Diagrams	19
3.3 Tools Used	21
3.4 Methodology	24

<b>4. TESTING AND RESULTS</b>	<b>26</b>
4.1 Testing	26
4.2 Results	27
4.2.1 Pest Detection	27
4.2.2 Pest Control	32
<b>5. CONCLUSION AND FUTURE SCOPE</b>	<b>34</b>
<b>BIBLIOGRAPHY</b>	<b>35</b>
<b>APPENDIX</b>	<b>36</b>
(A) Source Code	36
(B) User Manual	47

## **LIST OF TABLES**

<b>TABLE NO.</b>	<b>TABLE NAME</b>	<b>PAGE NO.</b>
1.	Literature Survey on Pest Detection and Control System using Deep Learning and IoT	15-16
2.	Test Cases of Pest Detection and Control System using Deep Learning and IoT	26

## **LIST OF FIGURES**

<b>FIGURE NO.</b>	<b>FIGURE NAME</b>	<b>PAGE NO.</b>
Figure 3.1.1	System Architecture of Pest Detection and Control System using Deep Learning and IoT	17
Figure 3.1.2	Circuit Diagram of Pesticide Sprayer Bot	18
Figure 3.2.1	Use Case Diagram of Pest Detection and Control System using Deep Learning and IoT	19
Figure 3.2.2	Sequence Diagram of Pest Detection and Control System using Deep Learning and IoT	20
Figure 3.3.1	Google Colab IDE	21

Figure 3.3.2	Arduino IDE	21
Figure 3.3.3	MIT App Inventor	22
Figure 3.3.4	LabelImg Tool	23
Figure 3.4.1	YOLO v5 (You Look Only Once version 5) Network Architecture	24
Figure 4.2.1.1	Camera detecting pest named aphid	27
Figure 4.2.1.2	Camera detecting pest named armyworm	27
Figure 4.2.1.3	Camera detecting pest named beetle	28
Figure 4.2.1.4	Camera detecting pest named bollworm	28
Figure 4.2.1.5	Camera detecting pest named grasshopper	29
Figure 4.2.1.6	Camera detecting pest named mite	29
Figure 4.2.1.7	Camera detecting pest named mosquito	30
Figure 4.2.1.8	Camera detecting pest named sawfly	30



Figure 4.2.1.9	Camera detecting pest named stem borer	31
Figure 4.2.2.1	Pesticide Sprayer Bot	32
Figure 4.2.2.2	Pesticide Sprayer Mobile App	33

## **ABSTRACT**

Agricultural and horticultural crops are attacked by a wide variety of pests, the most important being insects, mites, nematodes and gastropod mollusks. The damage they do results both from the direct injury they cause to the plants and from the indirect consequences of the fungal, bacterial or viral infections they transmit. The pests affecting trees are predominantly insects, and many of these have also been introduced inadvertently and lack natural enemies, and some have transmitted novel fungal diseases with devastating results. In existing systems of pest control, Humans have traditionally performed pest control in agriculture and forestry by the use of pesticides. But pesticides should not always be used the first time you see a pest. Sometimes pest problems may clear up on their own due to environmental factors and natural control. Additionally, pesticide spraying needs expertise and while spraying different pesticides, pesticide fumes might affect human's health. For this reason, automation in pest spraying is required. With massive advancement in technology, this is possible using Deep Learning and various sensors used in IoT.

In the proposed work , using Deep Learning algorithm – YOLO (You Only Look Once) we can recognize using a camera whether the insects are harmful or not, using a trained model on pest images. After pest image classification, the IoT module is immediately started and then the pesticide is sprayed. Precise position spraying can effectively reduce the amount of pesticides used and reduce pesticide damage to the soil.

# 1. INTRODUCTION

The industrialization of the agricultural sector has increased the chemical burden on natural ecosystems. Pesticides are agrochemicals used in agricultural lands, public health programs, and urban green areas in order to protect plants and humans from various diseases. However, due to their known ability to cause a large number of negative health and environmental effects, their side effects can be an important environmental health risk factor. The urgent need for a more sustainable and ecological approach has produced many innovative ideas, among them agriculture reforms and food production implementing sustainable practice evolving to food sovereignty. It is more obvious than ever that the society needs the implementation of a new agricultural concept regarding food production, which is safer for man and the environment.

Pesticides are substances or mixtures of substances that are mainly used in agriculture or in public health protection programs in order to protect plants from pests, weeds or diseases, and humans from vector-borne diseases, such as malaria, dengue fever, and schistosomiasis. Many of the pesticides in high concentrations have been associated with health and environmental issues. Exposure to pesticides can be through contact with the skin, ingestion, or inhalation. The numerous negative health effects that have been associated with chemical pesticides include, among other effects, dermatological, gastrointestinal, neurological, carcinogenic, respiratory, reproductive, and endocrine effects. Furthermore, high occupational, accidental, or intentional exposure to pesticides can result in hospitalization and death. Heavy treatment of soil with pesticides can cause populations of beneficial soil microorganisms to decline. Heavy pesticide spraying can directly hit non-target vegetation, or can drift or volatilize from the treated area and contaminate air, soil, and non-target plants. Some pesticide drift occurs during every application, even from ground equipment. Drift can account for a loss of 2 to 25% of the chemical being applied, which can spread over a distance of a few yards to several hundred miles. This pesticide drift can indirectly affect the distant plants and animals which were a few yards to several hundred miles apart.

This brings the urgent need for a new concept in agriculture involving a drastic reduction in the use of chemical pesticides. The reduction in the use of agrochemicals by applying them only when and where they are necessary, the spatiotemporal variability of all the soil and crop factors of a given field must be taken into consideration. This variability includes yield, field,

soil, and crop variability but also factors, such as wind damage or flooding. Technological systems, such as geographical information systems, global positioning systems, and various sensors, can be useful. These technological systems are developed by precision agriculture and the selected technological tools can be used to decrease risks for environmental pollution and water pollution and to enhance economic benefits stemming from the reduction in the use of chemical products.

### **1.1 Problem Definition**

To create a smart pesticide sprayer using Deep Learning and IoT such that humans are not affected, by reducing the amount of pesticide required to kill the pest, thus affectively spraying correct proportions of pesticide for the recognized type of pest. The pest is first recognized using a camera which is then classified using a trained model on Deep Learning Algorithm, YOLO (You Only Look Once). This project also helps in reducing environmental pollution and thus reduces pesticide concentration in air by using precision spraying.

### **1.2 Existing System**

The existing system gives different Deep Learning strategies in detecting pests only. There are no pest control mechanisms. Manuel pest spraying by humans is still in practice and the Deep Learning algorithms only gives us a part of real-time application for detecting the pests and spraying the pesticide.

### **1.3 Proposed System**

In the proposed system, the pests are being detected as well as a mechanism for pest control is provided. The mechanism proposed include – calculated quantity of pesticide spraying for the recognized pest and use of Deep Learning techniques not only for pest detection but also for pest localization.

## 1.4 System Requirements

### 1.4.1 Software Requirements:

Operating System	:	Windows XP/7/10
Coding Language	:	Python, C++
IDE	:	Google Colab, Arduino IDE

### 1.4.2 Hardware Requirements:

System	:	Intel/AMD 64 bit, Arduino Uno.
GPU	:	NVIDIA GPU driver version 450.36.06 or higher
RAM	:	8 GB

## 2. LITERATURE SURVEY ON PEST DETECTION AND CONTROL SYSTEM USING DEEP LEARNING AND IoT

The following literature reviews discuss about various pest detection systems.

**Ching-Ju Chen, Ya-Yu Huang, Yuan-Shuo Li, Ying-Cheng Chen, Chuan-Yu Chang, Yueh-Min Huang, 2021,** *Tessaratomia papillosa* (Drury) first invaded Taiwan in 2009. Every year, *T. papillosa* causes severe damage to the longan crops. Novel applications for edge intelligence are applied in this study to establish an intelligent pest recognition system to manage this pest problem. A detecting drone is used to photograph the pest and employed a Tiny-YOLOv3 neural network model built on an embedded system NVIDIA Jetson TX2 to recognize *T. papillosa* in the orchard to determine the position of the pests in real-time. The pests' positions are then used to plan the optimal pesticide spraying route for the agricultural drone. Apart from planning the optimized spraying of pesticide for the spraying drone, the TX2 embedded platform also transmits the position and generation of pests to the cloud to record and analyze the growth of longan with a computer or mobile device. This study enables farmers to understand the pest distribution and take appropriate precautions in real-time. The agricultural drone sprays pesticides only where needed, which reduces pesticide use, decreases damage to the environment, and increases crop yield.

**Rui Li, Rujing Wang, Jie Zhang, Chengjun Xie, Liu Liu, Fangyuan Wang, Hongbo Chen, Tianjiao Chen, Haiying Hu, Xiufang Jia, Min Hu, Man Zhou, Dengshan Li, Wancai Liu, 2019,** In agriculture, pest always causes the major damage in fields and results in significant crop yield losses. Currently, manual pest classification and counting are very time-consuming and many subjective factors can affect the population counting accuracy. In addition, the existing pest localization and recognition methods based on Convolutional Neural Network (CNN) are not satisfactory for practical pest prevention in fields because of pests' different scales and attitudes. In order to address these problems, an effective data augmentation strategy for CNN-based method is proposed in this paper. In training phase, we adopt data augmentation through rotating images by various degrees followed by cropping into different grids. In this way, we could obtain a large number of extra multi-scale examples that could be adopted to train a multi-scale pest detection model. In terms of test phase, the test time augmentation (TTA) strategy is utilized that separately inferences input images with various resolutions using

the trained multi-scale model. Finally, these detection results were fused from different image scales by non-maximum suppression (NMS) for the final result. Experimental results on wheat sawfly, wheat aphid, wheat mite and rice planthopper in our domain specific dataset, show that our proposed data augmentation strategy achieves the pest detection performance of 81.4% mean Average Precision (mAP), which improves 11.63%, 7.93%, 4.73% compared to three state-of-the-art approaches.

**J. Priyadarsini, B. Naveen Karthick, K. Karthick, B. Karthikeyan, S. Mohan, 2019,** The paper represents a smart device to show the harmful effects of pest bugs in the harvesting land and pH level in the water bodies. Different frequencies of ultrasonic waves are produced by this which are very helpful to repel different types of insects. To reduce the cost spent on pesticides. The great reduction of bugs can be noted if this device is implemented in harvesting land and we can avoid using pesticides.

**M. Manoj Vihari, Dr. Usha Rani. Nelakuditi, M. Purna Teja, 2018,** In India, agriculture plays a pivotal role and provides a principal means of livelihood. Pesticides save the crop from pests and improve the yield. At present pesticides and fertilizers are sprayed manually, which affect the human nervous system and causes many deaths every year. WHO declared there are more than one million pesticide cases present every year. This paper explains about remote controlled drone based sprayer system used in precision agriculture which avoids the direct spraying by humans. This system has advantage of reducing labor workers, spraying time and resources like water and chemicals over conventional spraying methods and also can improve yield and crop health.

**Preetha Rajan, Radhakrishnan B., Dr. L. Padma Suresh, 2016,** Agriculture is the mother of all culture. Economy and prosperity of a country depends on agriculture production. Agriculture provides food as well as raw material for industry. Agriculture production is inversely affected by pest infestation and plant diseases. Early pest identification and disease detection will help to minimize the loss of production. Naked eye observation is a common using method to identify the pest. But it is time consuming process. In this paper an automatic pest identification system using image processing techniques was proposed. Color feature is used to train the SVM to classify the pest pixels and leaf pixels. Morphological operations are used to remove the unwanted elements in the classified image.

**Table 1:** Literature survey on Pest Detection and Control System using Deep Learning and IoT

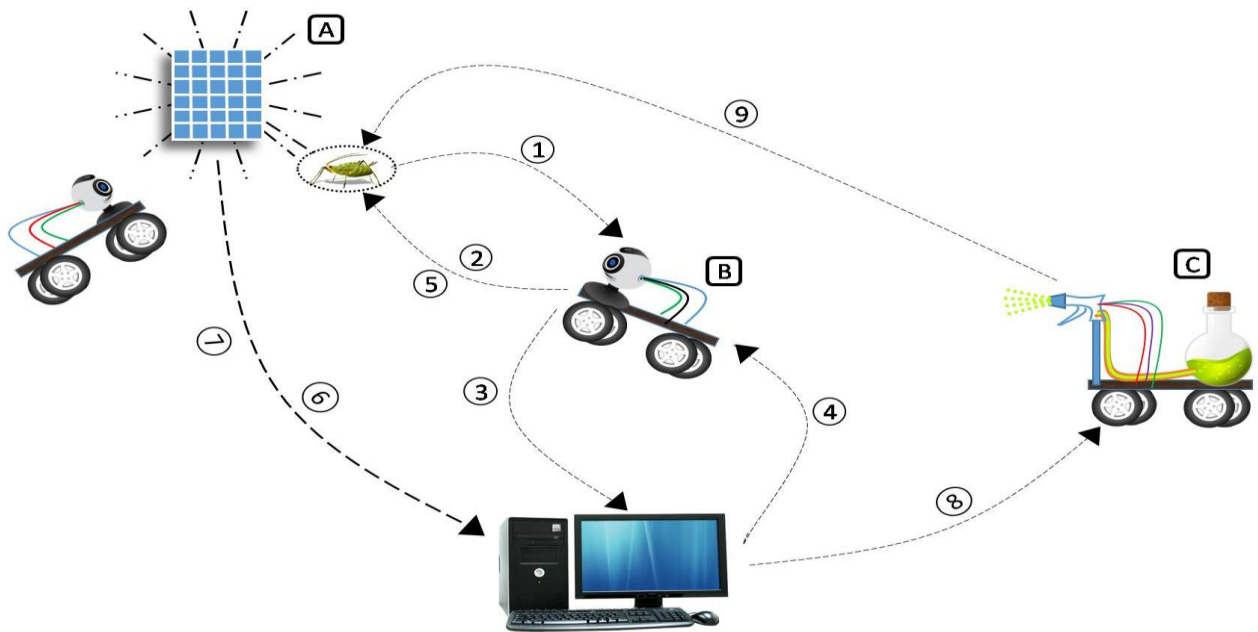
S No.	Year of Publication	Title	Algorithm used	Advantages	Disadvantages
1.	2021	Identification of Fruit Tree Pests With Deep Learning on Embedded Drone to Achieve Accurate Pesticide Spraying	YOLOv3 , Tiny YOLO v3	1. This work is useful for annihilating large swarm of pests. 2. In this work, path optimization methods were introduced for saving the drone's battery life.	1. Drone should be taken care from crashing into trees. 2. Only one type of pest was classified.
2.	2019	An Effective Data Augmentation Strategy for CNN-Based Pest Localization and Recognition in the Field	CNN	An effective data augmentation in training phase is proposed, which is feasible to apply for practical pest prevention.	1. CNN deep learning model takes lot of training time and energy. 2. No mechanism was implemented for pest control.
3.	2019	Detection of pH value and Pest control for ecofriendly agriculture	No Algorithms were used	1. Further enhances the scope of keeping pesticide concentration in check by using Soil pH value. 2. Gives an idea about organic farming	1. No algorithm implementation was proposed on pest control. 2. Too simplistic view on pesticide reduction



4.	2018	IoT based Unmanned Aerial Vehicle system for Agriculture applications	IoT Sensors and Motors were used	<p>1. Reduction in spraying time.</p> <p>2. It reduces the amount of spray.</p>	<p>1. No algorithms were used for classifying the pests and hence the drone should be loaded manually with pesticide when pest is detected.</p>
5.	2016	Detection And Classification Of Pests From Crop Images Using Support Vector Machine	Support Vector Machine	<p>1. Fast Classification with use of Machine Learning algorithm – SVM.</p> <p>2. Efficient in classifying large variety of pests</p>	<p>1. Sometimes the shadows will occur in the images or light may be reflected from the leaf. In that case the identification system cannot produce the correct results</p> <p>2. There are large number of pests. So it is very difficult to perform the classification operation at very high speed.</p>

### 3. DESIGN AND METHODOLOGY OF PEST DETECTION AND CONTROL SYSTEM USING DEEP LEARNING AND IoT

#### 3.1.1 System Architecture



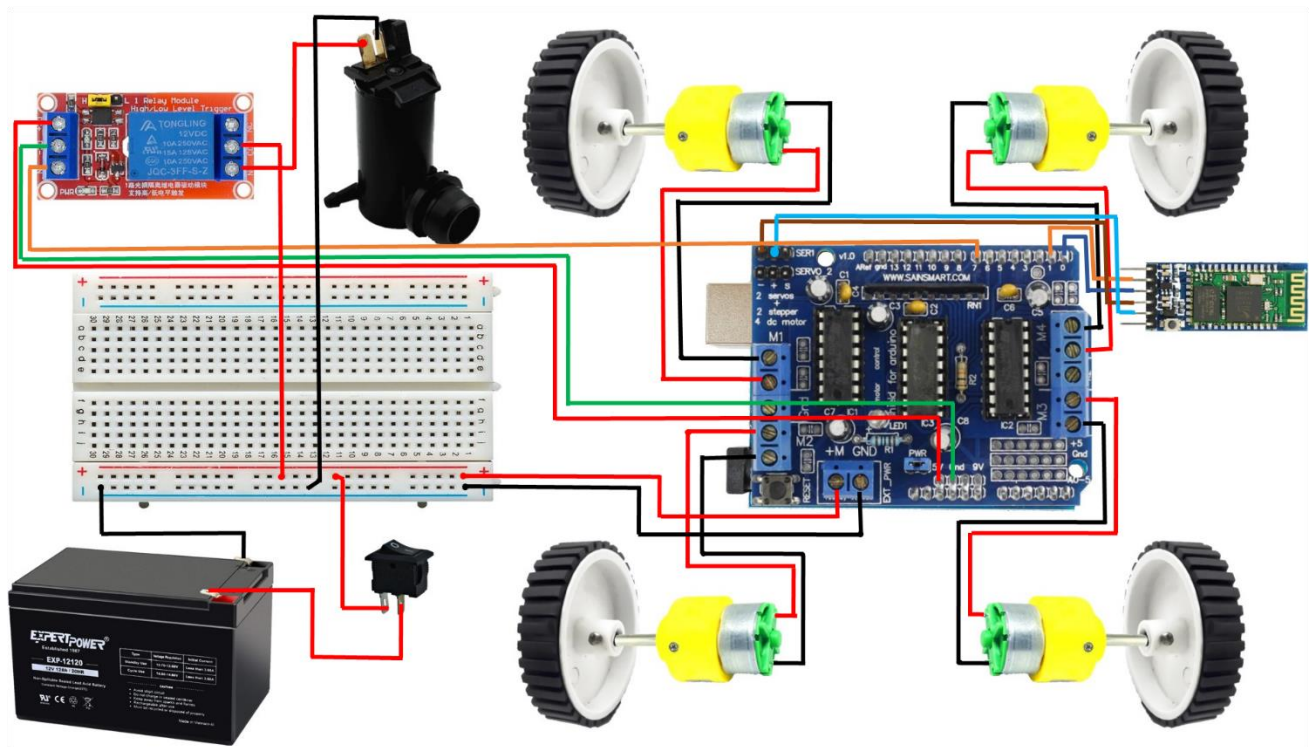
Symbol	Description
①	Rough location of pest sensed through temperature and humidity sensor
②	Approach the pest location and capture the pest image
③	Pest image sent for processing to computer
④	Pest image class detected
⑤	Accurately detect the pest and its location
⑥	Update the pest class and its location in computer
⑦	Move to next location of pest detected by sensor
⑧	Send pesticide sprayer bot to pest location on user demand
⑨	Spray the pesticide

Symbol	Description
A	Temperature and Humidity sensor
B	Wireless web-cam bot
C	Pesticide sprayer bot

**Figure 3.1.1:** System Architecture of Pest Detection and Control System using Deep Learning and IoT

The above **Figure 3.1.1** illustrates a conceptual model of the proposed system. Pests are detected using Web Camera. These images are sent to the computer, which already has a pre-trained Deep Learning model and thus classify the pests. The Pesticide Bot is sent to pest sighting location and the pesticide will be sprayed, based on the class of pest.

### 3.1.2 Pesticide Sprayer Bot Architecture

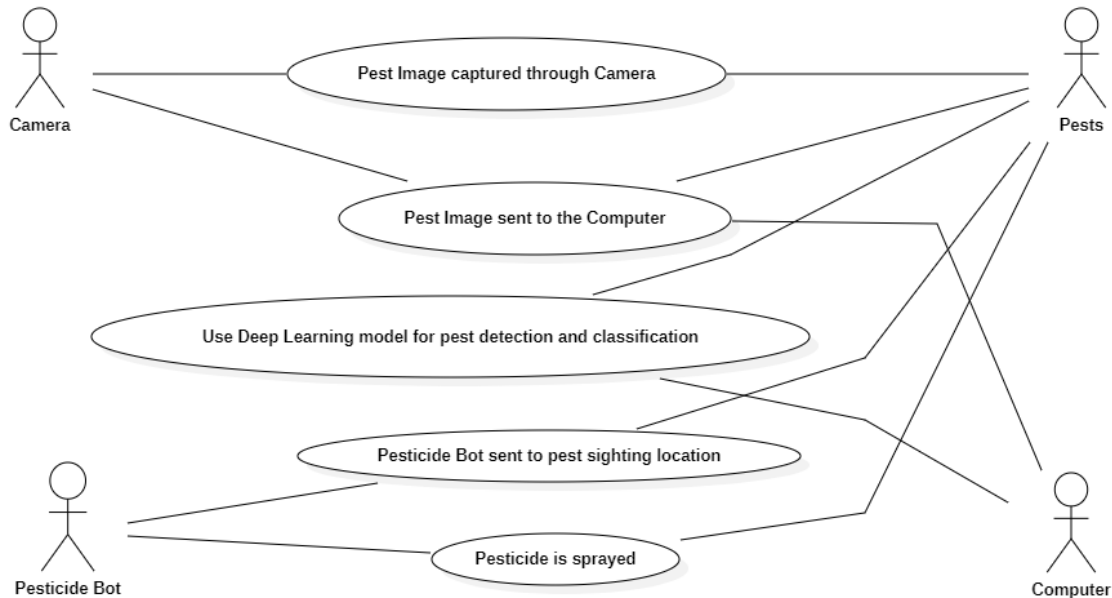


**Figure 3.1.2:** Circuit Diagram of Pesticide Sprayer Bot

The above **Figure 3.1.2** describes the design of Pesticide Sprayer Bot. Four wheels are connected to 12 volt DC motor of 150 rpm each. The motors are connected to L293N motor driver bridge which is placed on the Arduino Board. The motor driver is used to protect Arduino Board from high voltages. Here, 12v DC battery supply can damage the Arduino Board, but needed for driving the motors. A HC-05 Bluetooth module is connected to motor driver as shown above. The Bluetooth module is required for controlling the Pesticide Sprayer Bot movement and pesticide spraying. The input and output of Bluetooth signal is transmitted and received through 0, 1 pins of Arduino Board. A 5v 1-channel relay module is used, which acts as an electronic switch, to spray the pesticide. The input signal to the relay module is given from pin 7. On state of relay will spray the pesticide by closing the circuit that is connected with 12v submersible motor. Off state of relay will open the circuit and the pesticide spraying is stopped by disconnecting with submersible motor.

## 3.2 UML Diagrams

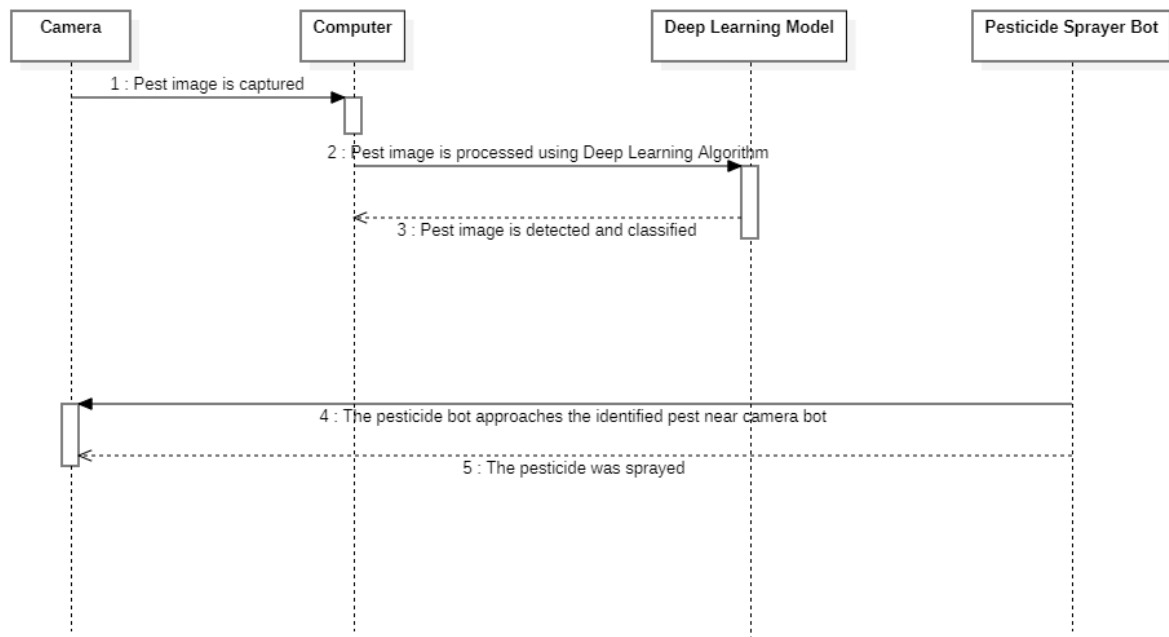
### Use Case Diagram:



**Figure 3.2.1:** Use Case Diagram of Pest Detection and Control System using Deep Learning and IoT

The above **Figure 3.2.1** describes how the actors in the system interacts with system components. The Camera as an Actor, first captures the pest image and transmit the image to the Computer on pest sighting. The Computer then uses Deep Learning Algorithm which was pre-trained before, to classify the pest image sent by the Camera. The decision whether the pesticide to be sprayed or not is decided by the user. When the decision to spray the pesticide is taken, the desired pesticide is sprayed on the recognized pest using the Pesticide Bot. Notice that the Pests as an Actor, it is indirectly interacting with the components based on pest sighting, its image being classified and also the pesticide being sprayed on it.

### Sequence Diagram:

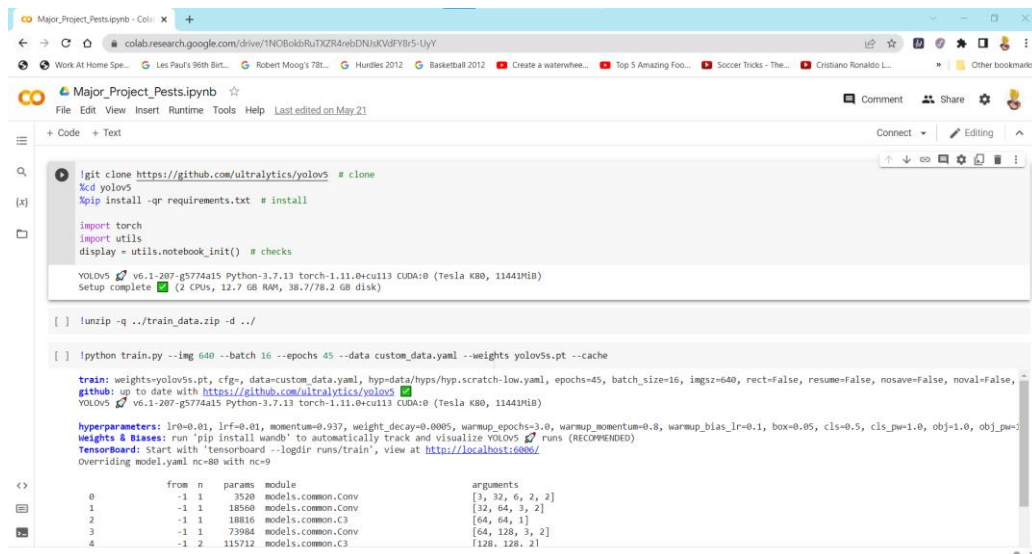


**Figure 3.2.2:** Sequence Diagram of Pest Detection and Control System using Deep Learning and IoT

The above diagram **Figure 3.2.2**, describes the order in which messages are shared among various components. The Camera first captures the pest image and the image is sent to the Computer. The received pest image is processed using Deep Learning Algorithm. After processing, the pest image is classified. The decision whether to spray the pesticide or not is sent to the Pesticide Sprayer Bot. The Pesticide Sprayer Bot sprays the required pesticide with a Pesticide Sprayer based on the results and decision.

## 3.3 Tools Used:

### 3.3.1 Google Colab:



```
git clone https://github.com/ultralytics/yolov5 # clone
%cd yolov5
%pip install -qr requirements.txt # install

import torch
import utils
display = utils.notebook_init() # checks

YOLOv5 v6.1.207-g5774a15 Python-3.7.13 torch-1.11.0+cu113 CUDA:0 (Tesla K80, 11441MiB)
Setup complete (2 CPUs, 12.7 GB RAM, 38.7/78.2 GB disk)

[ ] !unzip -q ../train_data.zip -d ../

[ ] !python train.py --img 640 --batch 16 --epochs 45 --data custom_data.yaml --weights yolov5s.pt --cache

train: weights=yolov5s.pt, cfg=, data=custom_data.yaml, hyp=data/hyps/hyp.scratch-low.yaml, epochs=45, batch_size=16, imgsz=640, rect=False, resume=False, nosave=False, noval=False,
github: up to date with https://github.com/ultralytics/yolov5
YOLOv5 v6.1.207-g5774a15 Python-3.7.13 torch-1.11.0+cu113 CUDA:0 (Tesla K80, 11441MiB)

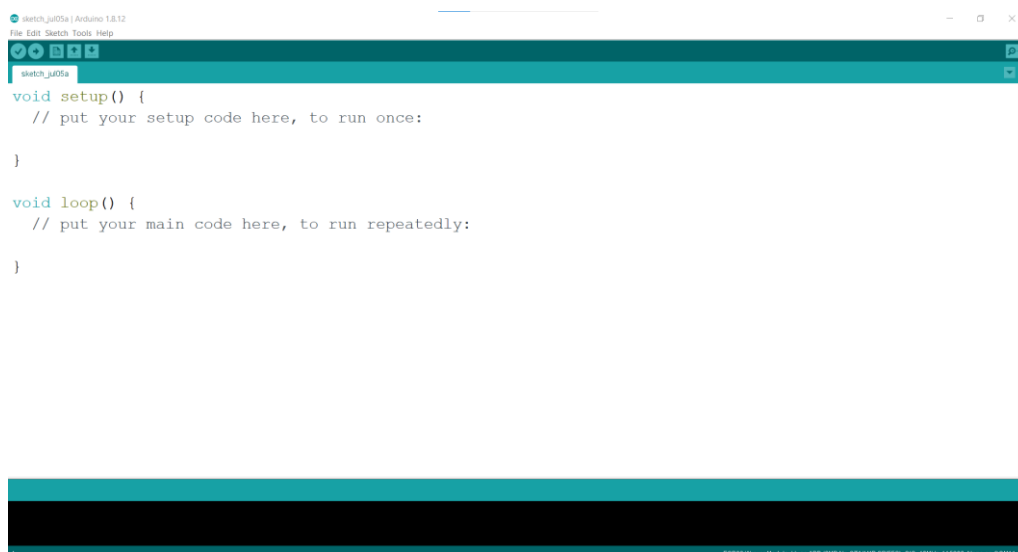
hyperparameters: lr0=0.01, lrf=0.01, momentum=0.937, weight_decay=0.0005, warmup_epochs=3.0, warmup_momentum=0.8, warmup_bias_lr=0.1, box=0.05, cls=0.5, cls_pw=1.0, obj=1.0, obj_pw=
Weights & Biases: run 'pip install wandb' to automatically track and visualize YOLOv5 runs (RECOMMENDED)
TensorBoard: Start with 'tensorboard --logdir runs/train', view at http://localhost:6006/
Overriding model.yaml nc=80 with nc=9

      from  n  params module                                arguments
      0      1    3520 models.common.Conv                  [3, 32, 6, 2, 2]
      1      1   18560 models.common.Conv                  [32, 64, 3, 2]
      2      1   18816 models.common.C3                     [64, 64, 1]
      3      1   73984 models.common.Conv                  [64, 128, 3, 2]
      4      2  115712 models.common.C3                     [128, 128, 2]
```

Figure 3.3.1: Google Colab IDE

The above **Figure 3.3.1** shows the Google Colab IDE. Colaboratory, or “Colab” for short, is a product from Google Research. Colab allows anybody to write and execute arbitrary python code through the browser, and is especially well suited to machine learning, data analysis and education. More technically, Colab is a hosted Jupyter notebook service that requires no setup to use, while providing access free of charge to computing resources including GPUs. Google servers providing the infrastructure to let the code run with a press of a button. In this project, Google Colab was used to train Deep Learning Model on pest dataset using YOLO-v5 algorithm. Google Colab IDE is available at: <https://colab.research.google.com>.

### 3.3.2 Arduino IDE:



```
sketch_ju05a | Arduino 1.8.12
File Edit Sketch Tools Help

sketch_ju05a

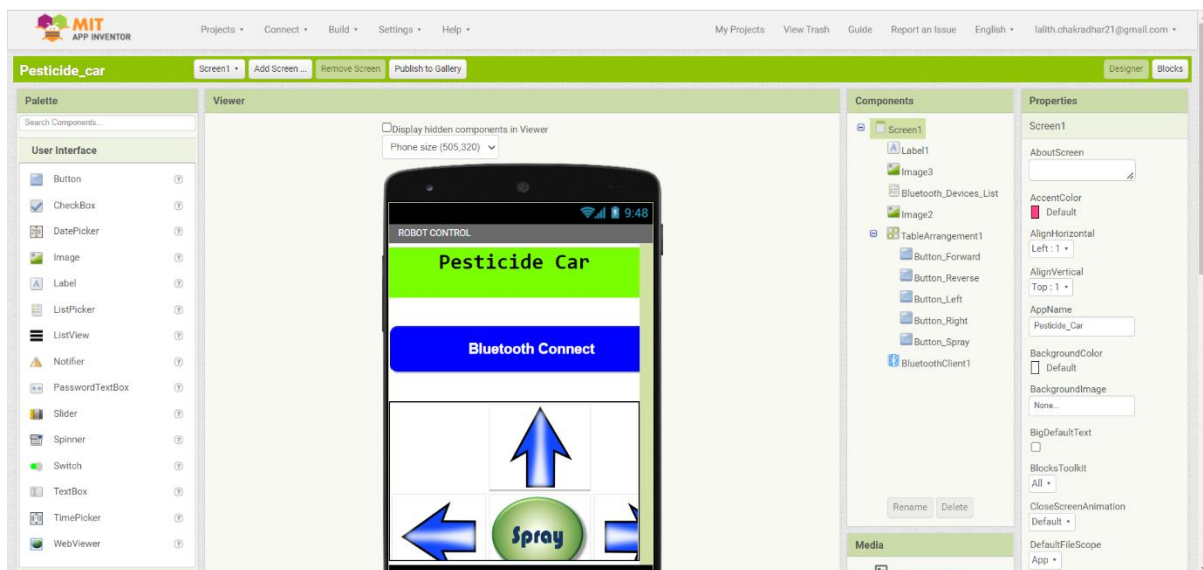
void setup() {
  // put your setup code here, to run once:
}

void loop() {
  // put your main code here, to run repeatedly:
}
```

Figure 3.3.2: Arduino IDE

The above **Figure 3.3.2** shows the Arduino IDE. Arduino is an open-source physical computing platform based on a simple I/O board and a development environment that implements the Processing/Wiring language. Arduino can be used to develop stand-alone interactive objects or can be connected to software on computer (e.g. Flash, Processing and MaxMSP). The boards can be assembled by hand or purchased preassembled. In this project, Arduino IDE was used to program the Pesticide Sprayer Bot. The open-source IDE can be downloaded for free at <https://arduino.cc>.

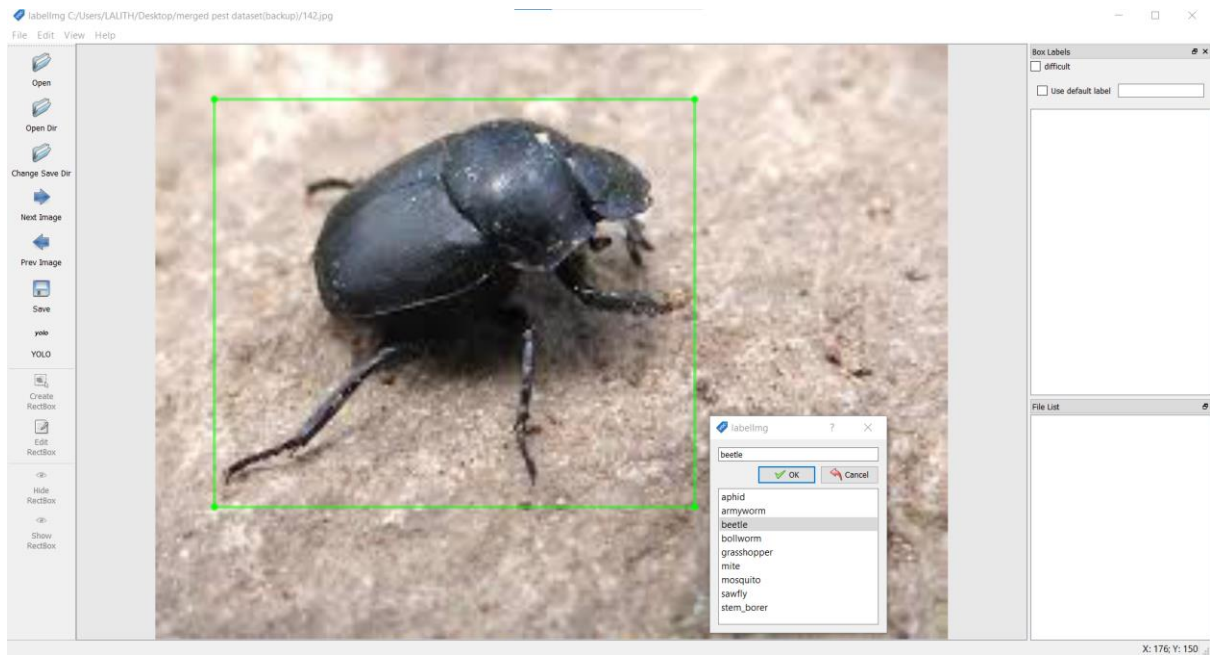
### 3.3.3 MIT App Inventor:



**Figure 3.3.3: MIT App Inventor**

The above **Figure 3.3.3** show the MIT App Inventor. MIT App Inventor is an intuitive, visual programming environment that allows everyone to build fully functional apps for Android phones, iPhones, and Android/iOS tablets. Those new to MIT App Inventor can have a simple first app up and running in less than 30 minutes. The blocks-based tool facilitates the creation of complex, high-impact apps in significantly less time than traditional programming environments. The MIT App Inventor project seeks to democratize software development by empowering all people, to move from technology consumption to technology creation. Blocks-based coding programs inspire intellectual and creative empowerment.

### 3.3.4 LabellImg:



**Figure 3.3.4:** LabellImg Tool

The above **Figure 3.3.4** shows the LabellImg tool. LabellImg is a graphical image annotation tool. It is written in Python and uses Qt for its graphical interface. Annotations are saved as XML files in PASCAL VOC format, the format used by ImageNet. Besides, it also supports YOLO and CreateML formats. LabellImg tool was used for creating bounding boxes and labelling pest classes for Deep Learning model's training data.

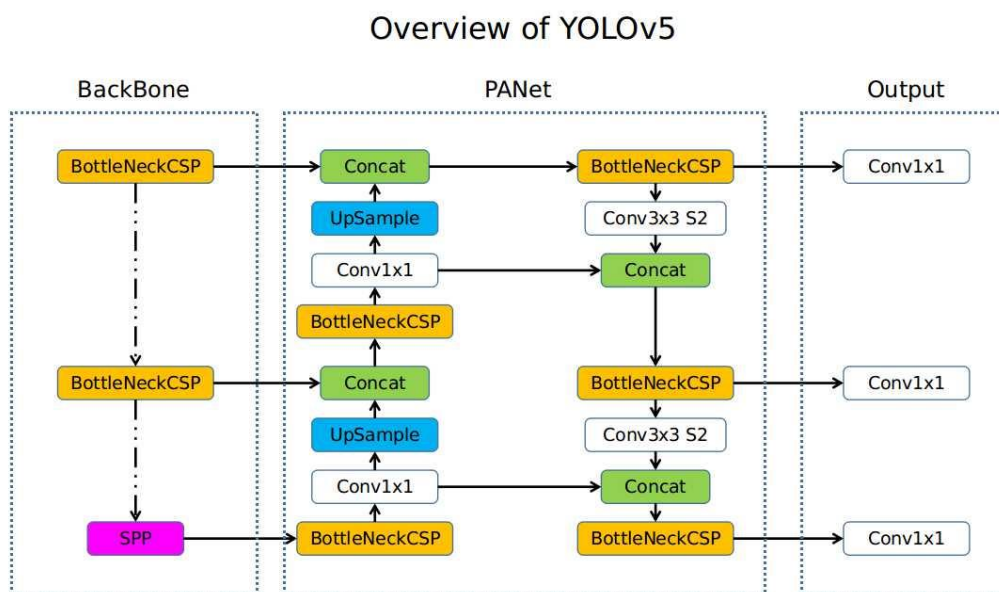


### 3.4 Methodology

The proposed system is executed in the following steps:

- A Camera is used for detecting the pest. A wireless web cam should be used.
- On a pest sighting, an image of the pest is captured and sent to the Computer.
- The Deep Learning Model is trained on pest images labelled with pest classes.
- The Computer uses Deep Learning Algorithm YOLO (You Look Only Once) to classify the pest.
- The image results are processed and the pest is now recognized. Notice that YOLO also gives the pest location using bounding boxes.
- The Pesticide Sprayer Bot is sent to Pest sighting location.
- The pesticide is sprayed at the location given by YOLO Algorithm.

#### YOLO Algorithm:



**Figure 3.4.1:** YOLO v5 (You Look Only Once version 5) Network Architecture

Source: <https://github.com/ultralytics/yolov5>

YOLO is a Convolutional Neural Network (CNN) for performing object detection in real-time. CNNs are classifier-based systems that can process input images as structured arrays of data and identify patterns between them (view image below). YOLO has the advantage of being much faster than other networks and still maintains accuracy. It allows the model to look at the whole image at test time, so its predictions are informed by the global context in the image.

YOLO and other convolutional neural network algorithms “score” regions based on their similarities to predefined classes. High-scoring regions are noted as positive detections of whatever class they most closely identify with. For example, in a live feed of traffic, YOLO can be used to detect different kinds of vehicles depending on which regions of the video score highly in comparison to predefined classes of vehicles.

The YOLOv5 algorithm first separates an image into a grid. Each grid cell predicts some number of boundary boxes (sometimes referred to as anchor boxes) around objects that score highly with the aforementioned predefined classes. Each boundary box has a respective confidence score of how accurate it assumes that prediction should be and detects only one object per bounding box. The boundary boxes are generated by clustering the dimensions of the ground truth boxes from the original dataset to find the most common shapes and sizes. YOLO v5 is different from all other prior releases, as this is a PyTorch implementation rather than a fork from original Darknet. YOLO v5 has a CSP backbone and PA-NET neck. The major improvements includes mosaic data augmentation and auto learning bounding box anchors.

Other comparable algorithms that can carry out the same objective are R-CNN (Region-based Convolutional Neural Networks made in 2015) and Fast R-CNN (R-CNN improvement developed in 2017), and Mask R-CNN. However, unlike systems like R-CNN and Fast R-CNN, YOLO is trained to do classification and bounding box regression at the same time.

## 4. TESTING AND RESULTS

### 4.1 Testing

Testing is the process of verifying and validating whether the application is bug-free and whether the technical requirements are met with the system design. Testing aims not only at finding faults in the existing software but also at finding measures to improve the software in terms of efficiency, accuracy, and usability. It mainly aims at measuring the specification, functionality, and performance of a software program or application.

Based on the code and hardware used in the system, following test cases were found:

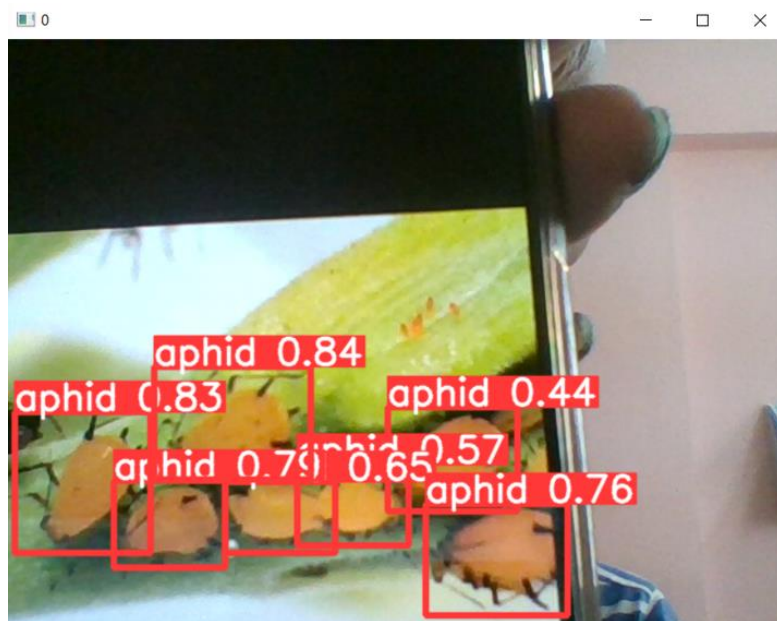
Test case number	Test case	Input	Expected output	Obtained output
1	The pest image captured was detected successfully and the pesticide was sprayed.	The pest image captured by the camera and the instruction given for pesticide bot to spray.	The pest should be eliminated by the pesticide.	The pest was successfully eliminated.
2	Camera captures the image of another insect or leaf.	The image captured by the camera.	No pest should be identified.	There was no bounding box on camera stream showing the identification of pest.
3	Pesticide droplets fell on Fertilizer Bot's electronic components on Pest spraying.	Signal to spray the pesticide given by User.	Short-circuit of Fertilizer Bot.	The Fertilizer Bot could sustain some pesticide droplets. The electronic component is later covered with a box.

**Table 2:** Test Cases of Pest Detection and Control System using Deep Learning and IoT

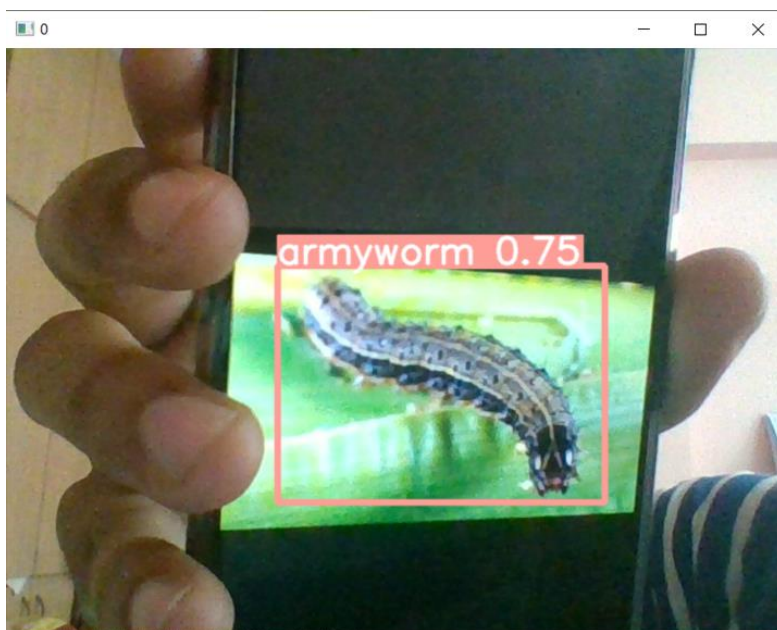
## 4.2 Results

### 4.2.1 Pest Detection:

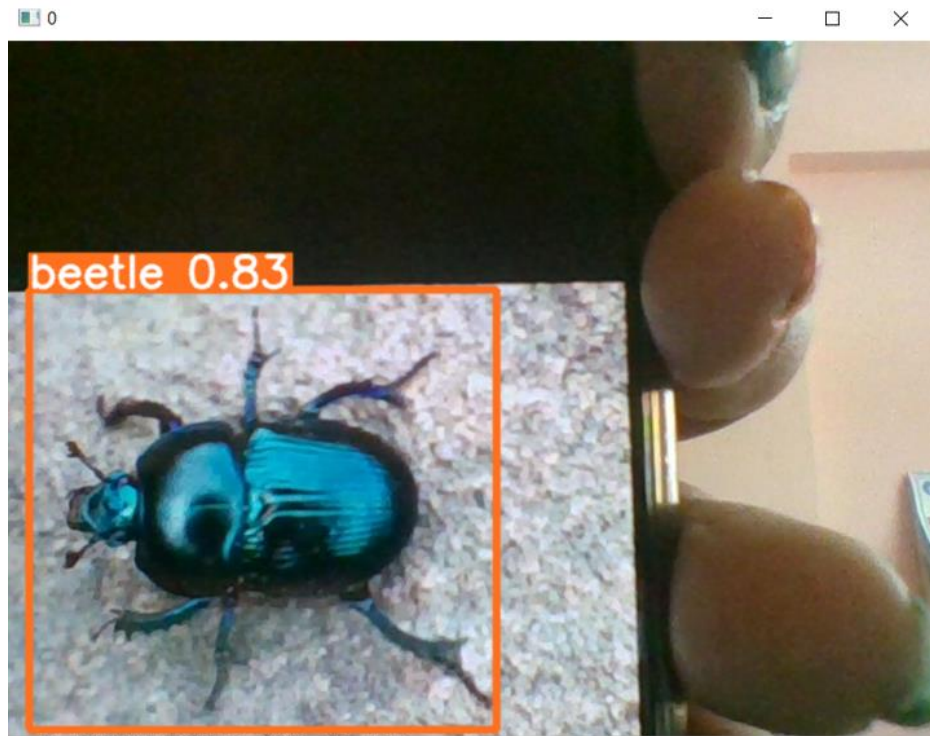
The Deep Learning model is trained on 555 pest images using YOLO-v5 algorithm. The trained model is able to detect the pests with 95% accuracy and can detect up to 9 pest classes. The pest classes include – aphid, armyworm, beetle, bollworm, grasshopper, mite, mosquito, stem borer.



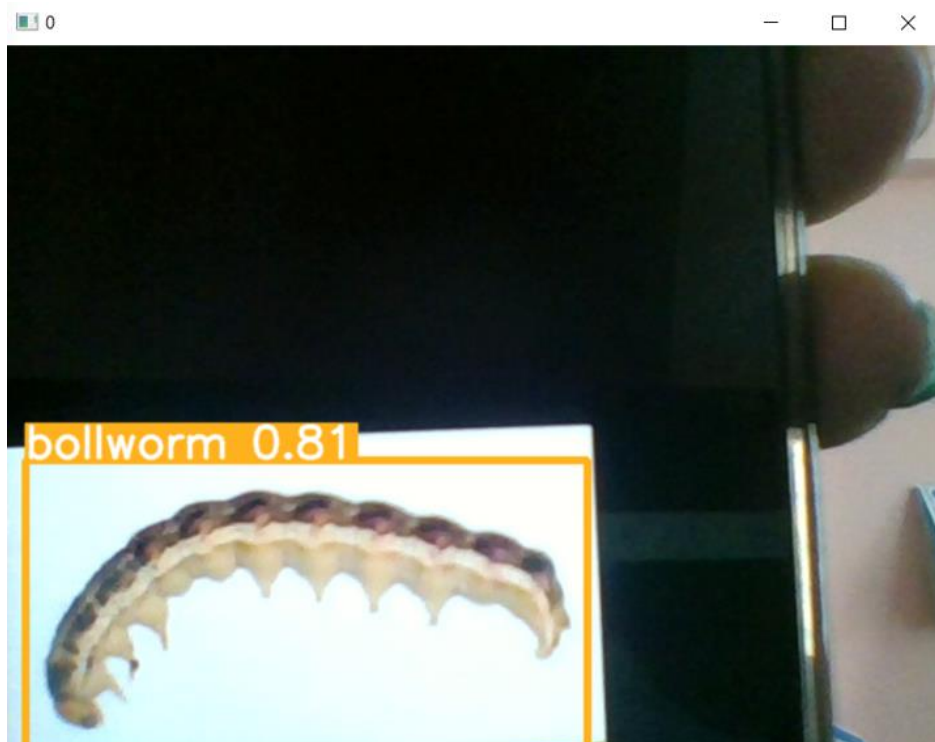
**Figure 4.2.1.1:** Camera detecting pest named aphid



**Figure 4.2.1.2:** Camera detecting pest named armyworm

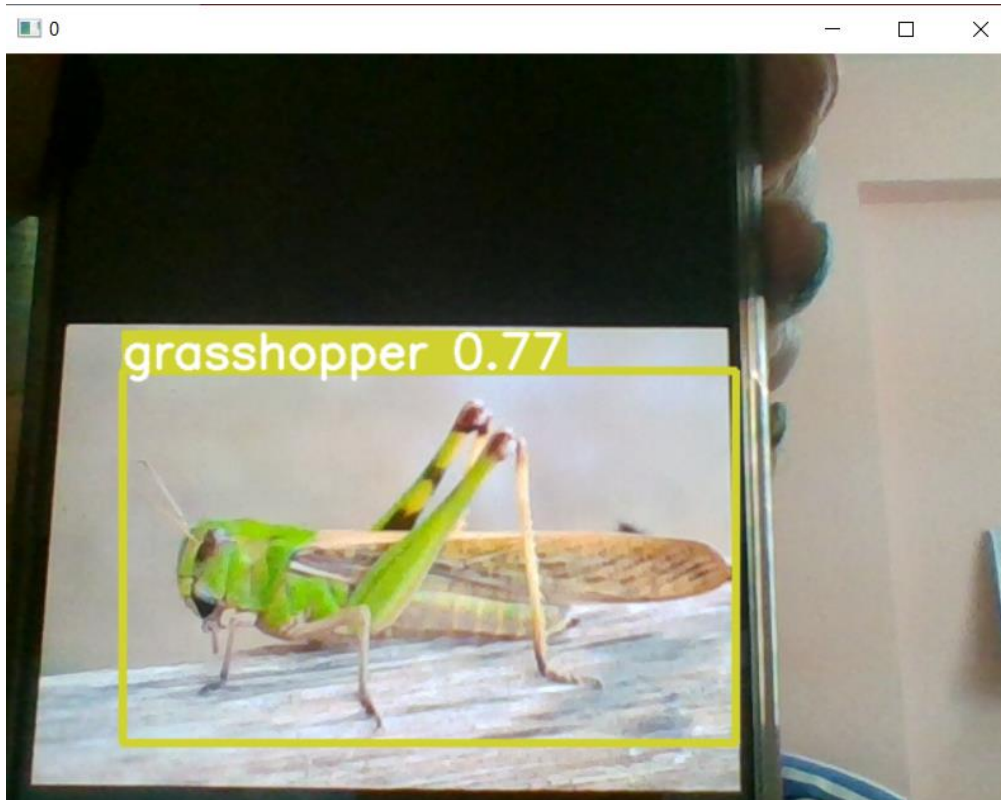


**Figure 4.2.1.3:** Camera detecting pest named beetle

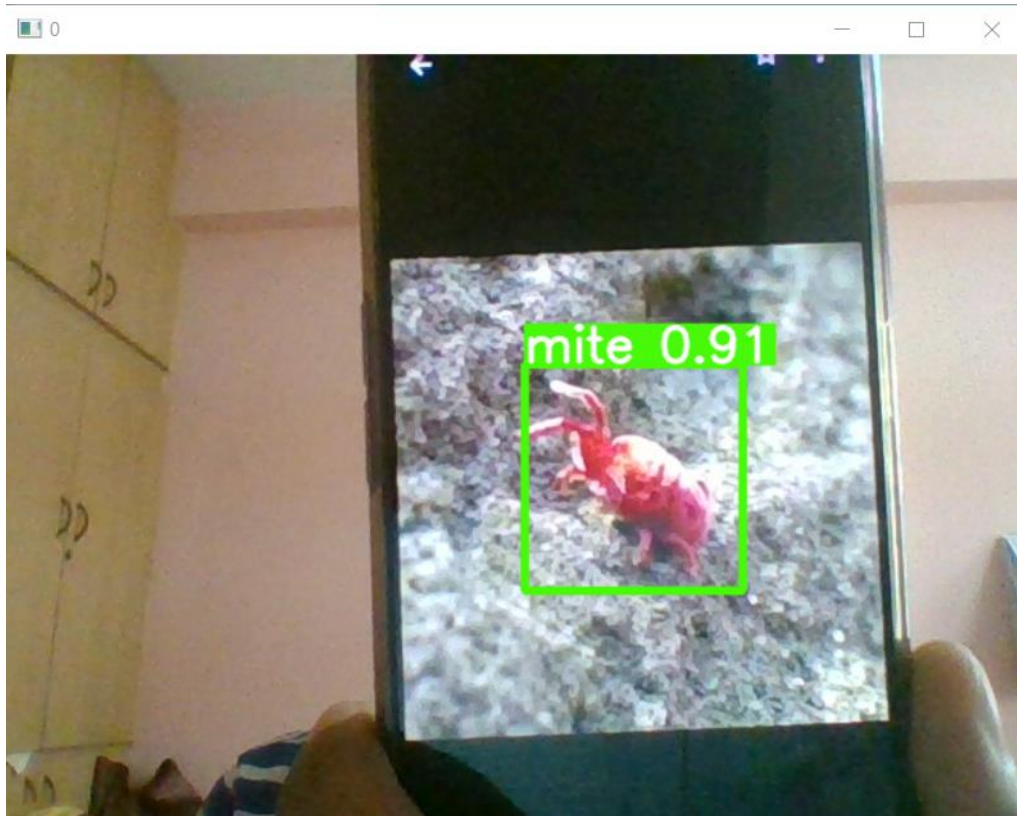


**Figure 4.2.1.4:** Camera detecting pest named bollworm

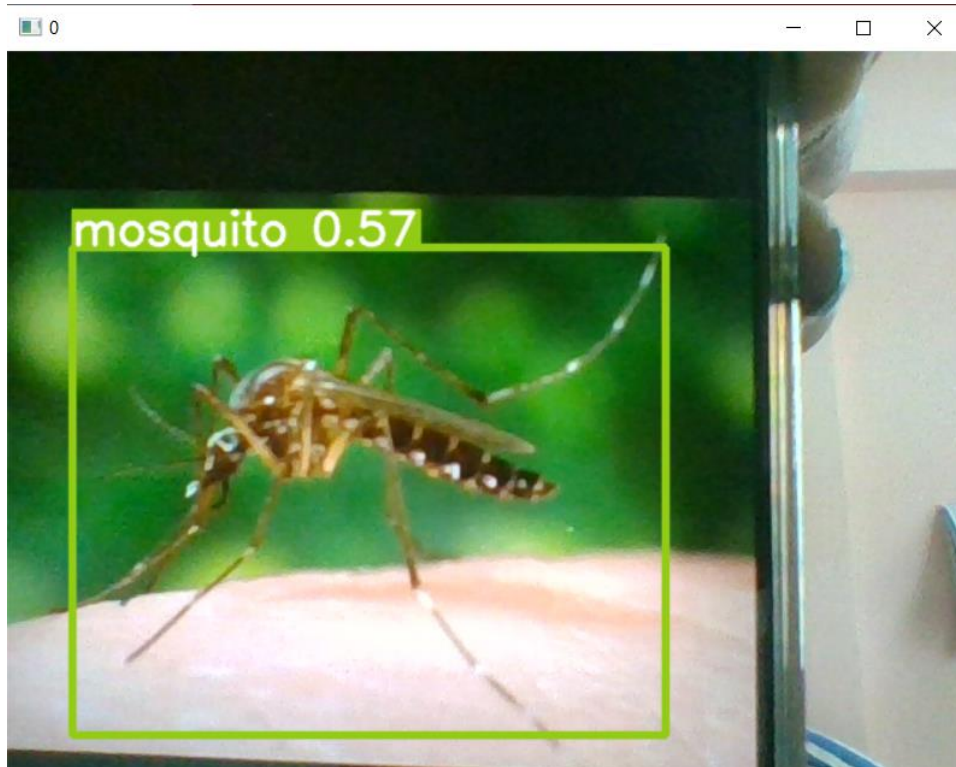




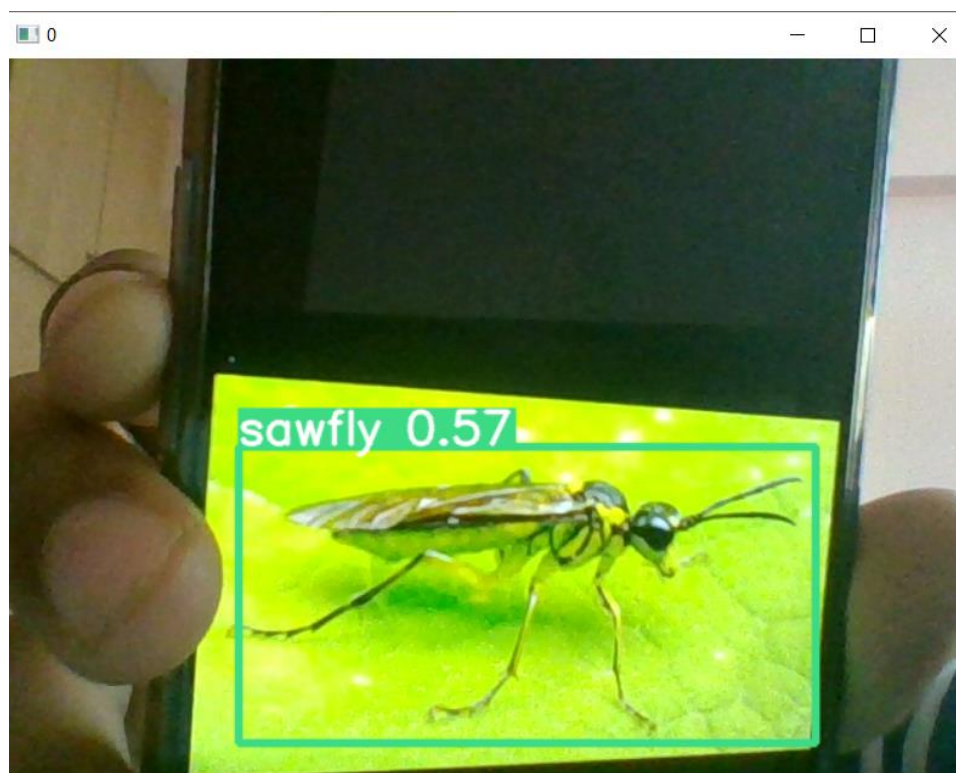
**Figure 4.2.1.5:** Camera detecting pest named grasshopper



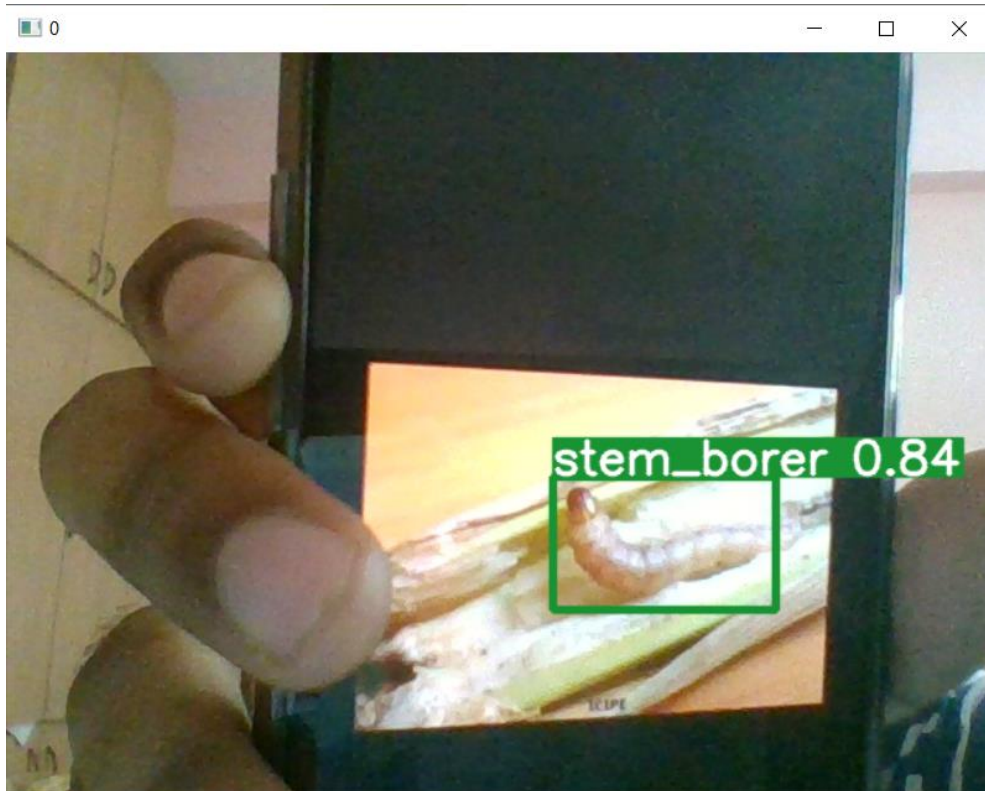
**Figure 4.2.1.6:** Camera detecting pest named mite



**Figure 4.2.1.7:** Camera detecting pest named mosquito



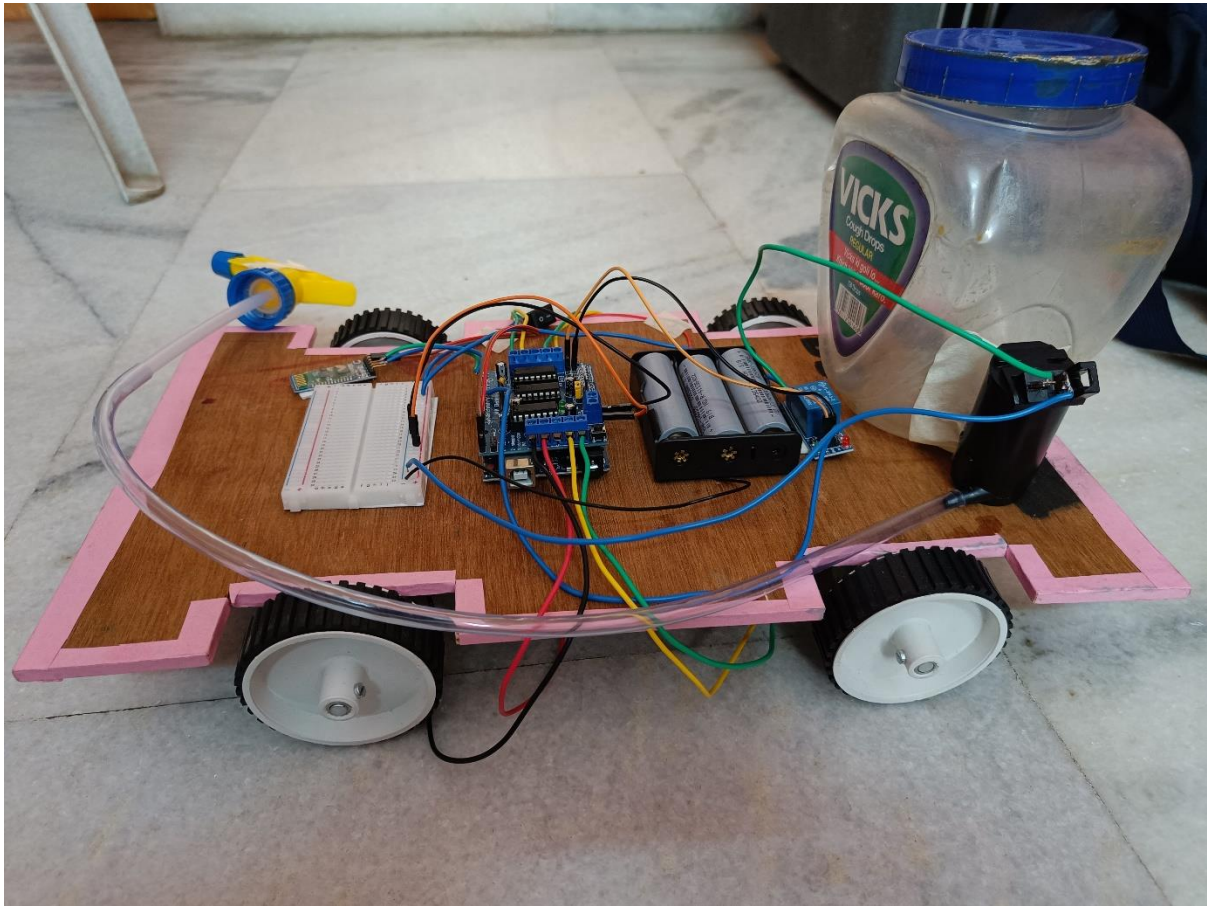
**Figure 4.2.1.8:** Camera detecting pest named sawfly



**Figure 4.2.1.9:** Camera detecting pest named stem borer

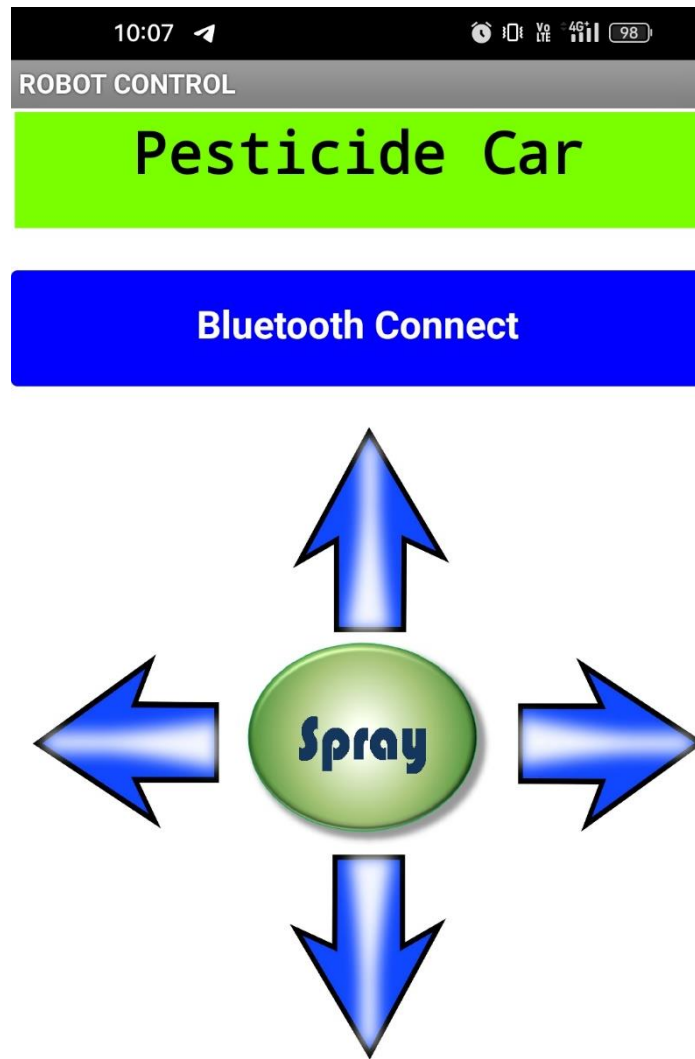


#### 4.2.2 Pest Control:



**Figure 4.2.2.1:** Pesticide Sprayer Bot

The above **Figure 4.2.2.1** is a picture of Pesticide Sprayer Bot. This bot is controlled using an app in phone via Bluetooth. The Pesticide Sprayer Bot is powered by 12v dc battery supply. The main purpose of this Bot is used to spray pesticide without any contact with human.



**Figure 4.2.2.2:** Pesticide Sprayer Mobile App

The above **Figure 4.2.2.2** shows the mobile app to control Pesticide Sprayer Bot. The app is created using MIT App Inventor. The up, down, left, right arrows are used to move the pesticide bot and the Spray button is used to spray the pesticide.

## **5. CONCLUSION AND FUTURE SCOPE**

### **Conclusion**

In this work, a mechanism to detect pest and control them is proposed. The latest version of YOLO algorithm (YOLO-v5) is very light weight and much accurate compared to its previous versions. The pesticide sprayer bot is easy to control and contact with pesticide fumes is avoided while spraying pesticide on the pests. Only limited amount of pesticide is sprayed each time and at the exact location where the deep learning model created bounding boxes of pest images. This system is cost effective and at the same time efficient at its purpose.

### **Future Scope**

This work can be incorporated in agriculture, where pesticide should be sprayed on crops. This system is effective in spraying small amount of pesticide and thus avoid over-use of pesticides. The farmers can stay far away from intoxicating pesticide fumes while spraying. On usage of best quality parts for IoT and create a Deep Learning model from scratch with deep layers and many weights assigned specifically to detect pests, a real time product can be launched with this work.

# BIBLIOGRAPHY

## Websites:

- 1) <https://www.arduino.cc/>
- 2) <https://colab.research.google.com/>
- 3) <https://appinventor.mit.edu/>

## References:

[1]: An Effective Data Augmentation Strategy for CNN-Based Pest Localization and Recognition in the Field

— Year published: 2019

— Authors: Rui Li, Rujing Wang, Jie Zhang, Chengjun Xie, Liu Liu, Fangyuan Wang, Hongbo Chen, Tianjiao Chen, Haiying Hu, Xiufang Jia, Min Hu, Man Zhou, Dengshan Li, Wancai Liu

[2]: Identification of Fruit Tree Pests with Deep Learning on Embedded Drone to Achieve Accurate Pesticide Spraying

— Year published: 2021

— Authors: Ching-Ju Chen, Ya-Yu Huang, Yuan-Shuo Li, Ying-Cheng Chen, Chuan-Yu Chang, Yueh-Min Huang

[3]: Detection of pH value and Pest control for eco-friendly agriculture

— Year published: 2019

— Authors: J. Priyadarsini, B. Naveen Karthick, K. Karthick, B. Karthikeyan, S. Mohan

[4]: IoT based Unmanned Aerial Vehicle system for Agriculture applications

— Year published: 2018

— Authors: M. Manoj Vihari, Dr. Usha Rani. Nelakuditi, M. Purna Teja

[5]: Detection And Classification Of Pests From Crop Images Using Support Vector Machine

— Year published: 2016

— Authors: Preetha Rajan, Radhakrishnan B., Dr. L. Padma Suresh

[6]: **Textbook** – Introduction to the IOT by IOT-OPEN.EU

[7]: **Textbook** - Hands-On Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems by Geron Aurelien

## APPENDIX

### (A) Source Code:

#### Deep Learning Model Code:

```
import argparse
import os
import sys
from pathlib import Path

import torch
import torch.backends.cudnn as cudnn

FILE = Path(__file__).resolve()
ROOT = FILE.parents[0] # YOLOv5 root directory
if str(ROOT) not in sys.path:
    sys.path.append(str(ROOT)) # add ROOT to PATH
ROOT = Path(os.path.relpath(ROOT, Path.cwd())) # relative

from models.common import DetectMultiBackend
from utils.dataloaders import IMG_FORMATS, VID_FORMATS, LoadImages, LoadStreams
from utils.general import (LOGGER, check_file, check_img_size, check_imshow,
                           check_requirements, colorstr, cv2,
                           increment_path, non_max_suppression, print_args, scale_coords,
                           strip_optimizer, xyxy2xywh)
from utils.plots import Annotator, colors, save_one_box
from utils.torch_utils import select_device, time_sync

@torch.no_grad()
def run(
```

```

weights=ROOT / 'yolov5s.pt', # model.pt path(s)
source=ROOT / 'data/images', # file/dir/URL/glob, 0 for webcam
data=ROOT / 'data/coco128.yaml', # dataset.yaml path
imgsz=(640, 640), # inference size (height, width)
conf_thres=0.25, # confidence threshold
iou_thres=0.45, # NMS IOU threshold
max_det=1000, # maximum detections per image
device="", # cuda device, i.e. 0 or 0,1,2,3 or cpu
view_img=False, # show results
save_txt=False, # save results to *.txt
save_conf=False, # save confidences in --save-txt labels
save_crop=False, # save cropped prediction boxes
nosave=False, # do not save images/videos
classes=None, # filter by class: --class 0, or --class 0 2 3
agnostic_nms=False, # class-agnostic NMS
augment=False, # augmented inference
visualize=False, # visualize features
update=False, # update all models
project=ROOT / 'runs/detect', # save results to project/name
name='exp', # save results to project/name
exist_ok=False, # existing project/name ok, do not increment
line_thickness=3, # bounding box thickness (pixels)
hide_labels=False, # hide labels
hide_conf=False, # hide confidences
half=False, # use FP16 half-precision inference
dnn=False, # use OpenCV DNN for ONNX inference
):
source = str(source)
save_img = not nosave and not source.endswith('.txt') # save inference images
is_file = Path(source).suffix[1:] in (IMG_FORMATS + VID_FORMATS)
is_url = source.lower().startswith(('rtsp://', 'rtmp://', 'http://', 'https://'))
webcam = source.isnumeric() or source.endswith('.txt') or (is_url and not is_file)
if is_url and is_file:
source = check_file(source) # download

```

```

# Directories
save_dir = increment_path(Path(project) / name, exist_ok=exist_ok) # increment run
(save_dir / 'labels' if save_txt else save_dir).mkdir(parents=True, exist_ok=True) # make
dir

# Load model
device = select_device(device)
model = DetectMultiBackend(weights, device=device, dnn=dnn, data=data, fp16=half)
stride, names, pt = model.stride, model.names, model.pt
imgsz = check_img_size(imgsz, s=stride) # check image size

# Dataloader
if webcam:
    view_img = check_imshow()
    cudnn.benchmark = True # set True to speed up constant image size inference
    dataset = LoadStreams(source, img_size=imgsz, stride=stride, auto=pt)
    bs = len(dataset) # batch_size
else:
    dataset = LoadImages(source, img_size=imgsz, stride=stride, auto=pt)
    bs = 1 # batch_size
vid_path, vid_writer = [None] * bs, [None] * bs

# Run inference
model.warmup(imgsz=(1 if pt else bs, 3, *imgsz)) # warmup
dt, seen = [0.0, 0.0, 0.0], 0
for path, im, im0s, vid_cap, s in dataset:
    t1 = time_sync()
    im = torch.from_numpy(im).to(device)
    im = im.half() if model.fp16 else im.float() # uint8 to fp16/32
    im /= 255 # 0 - 255 to 0.0 - 1.0
    if len(im.shape) == 3:
        im = im[None] # expand for batch dim
    t2 = time_sync()

```

```

dt[0] += t2 - t1

# Inference
visualize = increment_path(save_dir / Path(path).stem, mkdir=True) if visualize else False
pred = model(im, augment=augment, visualize=visualize)
t3 = time_sync()
dt[1] += t3 - t2

# NMS
pred = non_max_suppression(pred, conf_thres, iou_thres, classes, agnostic_nms,
max_det=max_det)
dt[2] += time_sync() - t3

# Second-stage classifier (optional)
# pred = utils.general.apply_classifier(pred, classifier_model, im, im0s)

# Process predictions
for i, det in enumerate(pred): # per image
    seen += 1
    if webcam: # batch_size >= 1
        p, im0, frame = path[i], im0s[i].copy(), dataset.count
        s += f'{i}: '
    else:
        p, im0, frame = path, im0s.copy(), getattr(dataset, 'frame', 0)

    p = Path(p) # to Path
    save_path = str(save_dir / p.name) # im.jpg
    txt_path = str(save_dir / 'labels' / p.stem) + (' if dataset.mode == 'image' else
f'_{frame}') # im.txt
    s += '%gx%g ' % im.shape[2:] # print string
    gn = torch.tensor(im0.shape)[[1, 0, 1, 0]] # normalization gain whwh
    imc = im0.copy() if save_crop else im0 # for save_crop
    annotator = Annotator(im0, line_width=line_thickness, example=str(names))
    if len(det):

```



```

# Rescale boxes from img_size to im0 size
det[:, :4] = scale_coords(im.shape[2:], det[:, :4], im0.shape).round()

# Print results
for c in det[:, -1].unique():
    n = (det[:, -1] == c).sum() # detections per class
    s += f"{n} {names[int(c)]}{'s' * (n > 1)}, " # add to string

# Write results
for *xyxy, conf, cls in reversed(det):
    if save_txt: # Write to file
        xywh = (xyxy2xywh(torch.tensor(xyxy).view(1, 4)) / gn).view(-1).tolist() #
normalized xywh
        line = (cls, *xywh, conf) if save_conf else (cls, *xywh) # label format
        with open(f'{txt_path}.txt', 'a') as f:
            f.write((' %g ' * len(line)).rstrip() % line + '\n')

    if save_img or save_crop or view_img: # Add bbox to image
        c = int(cls) # integer class
        label = None if hide_labels else (names[c] if hide_conf else f'{names[c]}
{conf:.2f}')
        annotator.box_label(xyxy, label, color=colors(c, True))
    if save_crop:
        save_one_box(xyxy, imc, file=save_dir / 'crops' / names[c] / f'{p.stem}.jpg',
BGR=True)

# Stream results
im0 = annotator.result()
if view_img:
    cv2.imshow(str(p), im0)
    cv2.waitKey(1) # 1 millisecond

# Save results (image with detections)
if save_img:

```

```

if dataset.mode == 'image':
    cv2.imwrite(save_path, im0)
else: # 'video' or 'stream'
    if vid_path[i] != save_path: # new video
        vid_path[i] = save_path
    if isinstance(vid_writer[i], cv2.VideoWriter):
        vid_writer[i].release() # release previous video writer
    if vid_cap: # video
        fps = vid_cap.get(cv2.CAP_PROP_FPS)
        w = int(vid_cap.get(cv2.CAP_PROP_FRAME_WIDTH))
        h = int(vid_cap.get(cv2.CAP_PROP_FRAME_HEIGHT))
    else: # stream
        fps, w, h = 30, im0.shape[1], im0.shape[0]
    save_path = str(Path(save_path).with_suffix('.mp4')) # force *.mp4 suffix on
results videos
    vid_writer[i] = cv2.VideoWriter(save_path, cv2.VideoWriter_fourcc(*'mp4v'),
fps, (w, h))
    vid_writer[i].write(im0)

# Print time (inference-only)
LOGGER.info(f'{s}Done. ({t3 - t2:.3f}s)')

# Print results
t = tuple(x / seen * 1E3 for x in dt) # speeds per image
LOGGER.info(f'Speed: %.1fms pre-process, %.1fms inference, %.1fms NMS per image at
shape {(1, 3, *imgsz)}' % t)
if save_txt or save_img:
    s = f"\n{len(list(save_dir.glob('labels/*.txt')))} labels saved to {save_dir / 'labels'}" if
save_txt else "
    LOGGER.info(f"Results saved to {colorstr('bold', save_dir)}{s}")
if update:
    strip_optimizer(weights) # update model (to fix SourceChangeWarning)

```

```

def parse_opt():
    parser = argparse.ArgumentParser()
    parser.add_argument('--weights', nargs='+', type=str, default=ROOT / 'yolov5s.pt',
help='model path(s)')
    parser.add_argument('--source', type=str, default=ROOT / 'data/images',
help='file/dir/URL/glob, 0 for webcam')
    parser.add_argument('--data', type=str, default=ROOT / 'data/coco128.yaml',
help='(optional) dataset.yaml path')
    parser.add_argument('--imgsz', '--img', '--img-size', nargs='+', type=int, default=[640],
help='inference size h,w')
    parser.add_argument('--conf-thres', type=float, default=0.25, help='confidence threshold')
    parser.add_argument('--iou-thres', type=float, default=0.45, help='NMS IoU threshold')
    parser.add_argument('--max-det', type=int, default=1000, help='maximum detections per
image')
    parser.add_argument('--device', default='', help='cuda device, i.e. 0 or 0,1,2,3 or cpu')
    parser.add_argument('--view-img', action='store_true', help='show results')
    parser.add_argument('--save-txt', action='store_true', help='save results to *.txt')
    parser.add_argument('--save-conf', action='store_true', help='save confidences in --save-txt
labels')
    parser.add_argument('--save-crop', action='store_true', help='save cropped prediction
boxes')
    parser.add_argument('--nosave', action='store_true', help='do not save images/videos')
    parser.add_argument('--classes', nargs='+', type=int, help='filter by class: --classes 0, or --
classes 0 2 3')
    parser.add_argument('--agnostic-nms', action='store_true', help='class-agnostic NMS')
    parser.add_argument('--augment', action='store_true', help='augmented inference')
    parser.add_argument('--visualize', action='store_true', help='visualize features')
    parser.add_argument('--update', action='store_true', help='update all models')
    parser.add_argument('--project', default=ROOT / 'runs/detect', help='save results to
project/name')
    parser.add_argument('--name', default='exp', help='save results to project/name')
    parser.add_argument('--exist-ok', action='store_true', help='existing project/name ok, do not
increment')

```

```

    parser.add_argument('--line-thickness', default=3, type=int, help='bounding box thickness
(pixels)')
    parser.add_argument('--hide-labels', default=False, action='store_true', help='hide labels')
    parser.add_argument('--hide-conf', default=False, action='store_true', help='hide
confidences')
    parser.add_argument('--half', action='store_true', help='use FP16 half-precision inference')
    parser.add_argument('--dnn', action='store_true', help='use OpenCV DNN for ONNX
inference')

    opt = parser.parse_args()
    opt.imgsz *= 2 if len(opt.imgsz) == 1 else 1 # expand
    print_args(vars(opt))
    return opt

def main(opt):
    check_requirements(exclude=('tensorboard', 'thop'))
    run(**vars(opt))

if __name__ == "__main__":
    opt = parse_opt()
    main(opt)

```

## IoT Code:

```
#include "AFMotor.h"

const int relayPin = 7;
const int MOTOR_1 = 1;
const int MOTOR_2 = 2;
const int MOTOR_3 = 3;
const int MOTOR_4 = 4;

AF_DCMotor motor1(MOTOR_1); // create motor object, 64KHz pwm
AF_DCMotor motor2(MOTOR_2); // create motor object, 64KHz pwm
AF_DCMotor motor3(MOTOR_3); // create motor object, 64KHz pwm
AF_DCMotor motor4(MOTOR_4); // create motor object, 64KHz pwm

int state;
int Speed = 250;

void setup() {
  motor1.setSpeed(Speed); // set the motor speed to 0-255
  motor2.setSpeed(Speed);
  motor3.setSpeed(Speed);
  motor4.setSpeed(Speed);
  pinMode(relayPin, OUTPUT); //define the relay signal pin
  digitalWrite(relayPin, HIGH); //initialize in off state
  Serial.begin(9600);
  delay(500);
}

void loop(){
  if(Serial.available() > 0){ //if some data is sent, reads it and saves in state
    state = Serial.read();
    if(state > 15){Speed = state;}
```

```

}

motor1.setSpeed(Speed);    // set the motor speed to 0-255
motor2.setSpeed(Speed);
motor3.setSpeed(Speed);
motor4.setSpeed(Speed);

//=====
//                Key Control Command
//=====

    if(state == 1){forward(); } // if the state is '1' the DC motor will go forward
    else if(state == 2){backward();} // if the state is '2' the motor will Reverse
    else if(state == 3){turnLeft();} // if the state is '3' the motor will turn left
    else if(state == 4){turnRight();} // if the state is '4' the motor will turn right
    else if(state == 5){Pump();}    // if the state is '5' the pesticide will be pumped
    else if(state == 6){discPump();} // if the state is '6' the pesticide will stop pumping
    else if(state == 7){Stop();}    //if the state is '7' the motor will stop
    ////////////////////////////////////END////////////////////////////////////

    delay(80);
}

void forward(){
    motor1.run(FORWARD); // turn it on going forward
    motor2.run(FORWARD);
    motor3.run(FORWARD);
    motor4.run(FORWARD);
}

void backward(){
    motor1.run(BACKWARD); // the other way
    motor2.run(BACKWARD);
    motor3.run(BACKWARD);

```

```
motor4.run(BACKWARD);  
}
```

```
void turnRight(){  
  motor1.run(FORWARD); // the other right  
  motor2.run(FORWARD);  
  motor3.run(BACKWARD);  
  motor4.run(BACKWARD);  
}
```

```
void turnLeft(){  
  motor1.run(BACKWARD); // turn it on going left  
  motor2.run(BACKWARD);  
  motor3.run(FORWARD);  
  motor4.run(FORWARD);  
}
```

```
void Pump(){  
  digitalWrite(relayPin, HIGH);  
  delay(3000);  
}
```

```
void discPump(){  
  digitalWrite(relayPin, LOW);  
}
```

```
void Stop(){  
  motor1.run(RELEASE); // stopped  
  motor2.run(RELEASE);  
  motor3.run(RELEASE);  
  motor4.run(RELEASE);  
}
```

## **(B) User Manuel:**

### **How to use Labelling tool for labelling Training Data:**

1. In data/predefined\_classes.txt define the list of classes that will be used for your training.
2. Right below "Save" button in the toolbar, click "PascalVOC" button to switch to YOLO format.
3. You may use Open/OpenDIR to process single or multiple images. When finished with a single image, click save.

A txt file of YOLO format will be saved in the same folder as your image with same name. A file named "classes.txt" is saved to that folder too. "classes.txt" defines the list of class names that your YOLO label refers to.

### **How to use Google Colab:**

1. Click on the NEW NOTEBOOK button to create a new Colab notebook. You can also upload your local notebook to Colab by clicking the upload button. You can rename your notebook by clicking on the notebook name and change it to anything you want.

#### **2. Google Colab Runtimes – Choosing the GPU or TPU Option**

- Click 'Runtime' on the top menu and select 'Change Runtime Type'.
- Here you can change the runtime according to your need.

#### **3. Using Terminal Commands on Google Colab**

- You can use the Colab cell for running terminal commands. Most of the popular libraries come installed by default on Google Colab. Yes, Python libraries like Pandas, NumPy, scikit-learn are all pre-installed.

- If you want to run a different Python library, you can always install it inside your Colab notebook like this:

```
!pip install library_name
```

Put an exclamation (!) before writing each command.



## **How to use MIT App Inventor:**

1. Go to App Inventor on the web.
2. Log in to App Inventor with a Gmail user name and password.
3. Start a new project. Type in the project name (underscores are allowed, spaces are not) and click OK.
4. You are now in the Designer, where you lay out the "user interface" of your app. The Design Window, or simply "Designer" is where you lay out the look and feel of your app, and specify what functionalities it should have. You choose things for the user interface things like Buttons, Images, and Text boxes, and functionalities like Text-to-Speech, Sensors, and GPS. For example – If your project needs a button. Click and hold on the word "Button" in the palette. Drag your mouse over to the Viewer. Drop the button and a new button will appear on the Viewer.
5. Switch over to the Blocks Editor. Blocks Editor is used to define actions to User Interface items. The Blocks Editor is where you program the behavior of your app. There are Built-in blocks that handle things like math, logic, and text. Below that are the blocks that go with each of the components in your app. In order to get the blocks for a certain component to show up in the Blocks Editor, you first have to add that component to your app through the Designer Editor. For example – If your project has a Button labelled as Button1 in Designer then switch to Blocks Editor. Click on the Button1 drawer. Click and hold the when Button1.Click do block. Drag it over to the workspace and drop it there. This is the block that will handle what happens when the button on your app is clicked. It is called an "Event Handler".
6. Connect App Inventor to your phone for live testing. Get the MIT AI2 Companion from the Play Store and install it on your phone or tablet. Start the AICompanion on your device
7. Get the Connection Code from App Inventor and scan or type it into your Companion app. See your app on the connected device.