

Intelligent Weed Removal System

by

Sattwik Ghatak 21BEC1373

Anusmita Panji 21BEC1381

Anusha Das 21BEC1388

Aayush Panwar 21BEC1609

A project report submitted to

Dr.VELMATHIG

SCHOOL OF ELECTRONICS ENGINEERING

In partial fulfillment of the requirements for the course of

BECE312L – Robotics & Automation

in

**B.Tech. ELECTRONICS AND COMMUNICATION
ENGINEERING**



Vandalur–Kelambakkam Road

Chennai – 600127

APRIL 2024

BONAFIDECERTIFICATE

It is certified that this project report entitled “**Smart Weeding Robotic Arm**” is a bonafide work of **Sattwik Ghatak(21BEC1373)**, **Anusmita Panji(21BEC1381)**, **Anusha Das(21BEC1388)** and **Aayush Panwar(21BEC1609)** who carried out the Project work under my supervision and guidance for BECE312L – Robotics & Automation.

Dr.VELMATHI-G

Professor Higher Academic Grade

School of Electronics Engineering (SENSE),

VIT University, Chennai

Chennai – 600 127.

ABSTRACT

The smart weeding robotic arm project aims to automate the process of clearing the field of unwanted plants (weed) by distinguishing between weeds and crops and then plucking them in agricultural fields. The project utilizes various components such as the NodeMCU ESP8266, robotic arm, servo motors, ESP32 CAM, and the MobileNet V2 for efficient image classification. To enable manual control over the robotic arm, the Blynk console is integrated into the system. This allows users to remotely control and manipulate the arm's movements, facilitating precise weed removal. Moreover, the project employs Edge Impulse, a powerful platform for developing machine learning models. The image classification model is trained using Edge Impulse and subsequently uploaded onto the ESP32 CAM module. This model enables the robotic arm to accurately differentiate between weeds and crops in real-time. By combining these technologies, the smart weeding robotic arm project presents an innovative solution for automating the labor-intensive task of weeding in agricultural settings. The integration of image classification and manual control ensures efficient and accurate weed removal, reducing the reliance on manual labor and improving overall crop yield.

Keywords: Blynk, Image Classification, Weed Plucking

ACKNOWLEDGEMENT

We wish to express our sincere thanks and deep sense of gratitude to our project guide, **Dr. VELMATHI G**, Professor Higher Academic Grade, School of Electronics Engineering, for her consistent encouragement and valuable guidance offered to us in a pleasant manner throughout the course of the project work.


We are extremely grateful to **Dr Susan Elias**, Dean of School of Electronics Engineering, VIT Chennai, for extending the facilities of the school towards our project and for her unstinting support.

We express our thanks to our Head of the ECE Department, **Dr. Mohanaprasad K** for his support rendered throughout the course of this project.

We also take this opportunity to thank all the faculty and staff of the school for their support and their wisdom imparted to us throughout the course.

We thank our parents, family, and friends for bearing with us throughout the course of our project and for the opportunity they provided us in undergoing this course in such a prestigious institution.

NAME WITH SIGNATURE:



Sattwik Ghatak Anusmita Panji Aayush Panwar Anusha Das

Table Of Contents:

S.NO	TITLE	PAGE.NO.
	ABSTRACT	3
	ACKNOWLEDGEMENT	4
1.	INTRODUCTION	6
	1.1 OBJECTIVES AND GOALS	6
	1.2 APPLICATIONS	6
	1.3 FEATURES	6
2.	IMPLEMENTATION	7
	2.1DESIGN AND IMPLEMENTATION	7-9
	2.2BLOCK DIAGRAM	9
	2.3HARDWARE ANALYSIS	10-13
	2.4(SNAPSHOTS-PROJECT,TEAM, RESULTS)	13
3.	3.1SOFTWARE–CODINGANDANALYSIS	15
	3.2SNALPSHOTS OF CODING AND RESULT	17
4.	INFERENCE,RESULT,AND CONCLUSION	19
5.	FUTURE WORK	22
6.	REFERENCES	24
7.	PHOTO GRAPH OF THE PROJECT ALONG WITH TEAM MEMBERS	25
8.	Project Video	26

1. INTRODUCTION

1.1 OBJECTIVES AND GOALS

- Develop a Intelligent Weed Removal System to automate and optimize weed management processes in agriculture.
- Implement IoT and AI technologies to enable efficient weed detection and targeted removal, aiming to enhance crop yield and reduce labor costs.
- Integrate edge computing capabilities to facilitate real-time decision-making and on-device inference for swift and accurate weed classification.
- Provide farmers with a reliable and cost-effective solution that promotes sustainable farming practices by minimizing herbicide usage and environmental impact.

1.2 APPLICATION

- Precision Weed Control: The system can be deployed in various agricultural settings to precisely identify and eliminate weeds while minimizing damage to crops.
- Remote Monitoring and Management: Farmers can remotely monitor and control the robotic arm's operations using a user-friendly interface, enabling efficient weed management without constant physical presence.
- Adaptive Deployment: The project's modular design allows for easy customization and adaptation to different crop types and field conditions, enhancing flexibility and scalability.

1.3 FEATURES

- Image Classification: Utilizing the MobileNet V2 model, the system can classify images captured by the onboard camera into weed and crop categories with high accuracy.
- Manual Control Interface: Integration with the Blynk console enables farmers to manually control the robotic arm's movements for targeted weed removal or override automated functions.
- Edge Computing: The deployment of lightweight image classification models on edge devices such as the ESP32-CAM enables fast and efficient inference directly on the device, reducing latency and bandwidth requirements.
- Energy-Efficient Design: The project prioritizes energy efficiency, utilizing low-power components and optimizing algorithms to prolong battery life and minimize operational costs, ideal for remote or off-grid farming applications.

2.IMPLEMENTATION

2.1 DESIGN AND IMPLEMENTATION:

ESP8266 to Servo Motors

Connections,

GND → GND

3v3 → V_{cc}

GPIO 16 → PWM pin

GND → GND

3v3 → V_{cc}

GPIO 4 → PWM pin

GND → GND

3v3 → V_{cc}

GPIO 2 → PWM pin

GND → GND

3v3 → V_{cc}

GPIO 5 → PWM pin

Esp-32to FTDI:

GND → GND

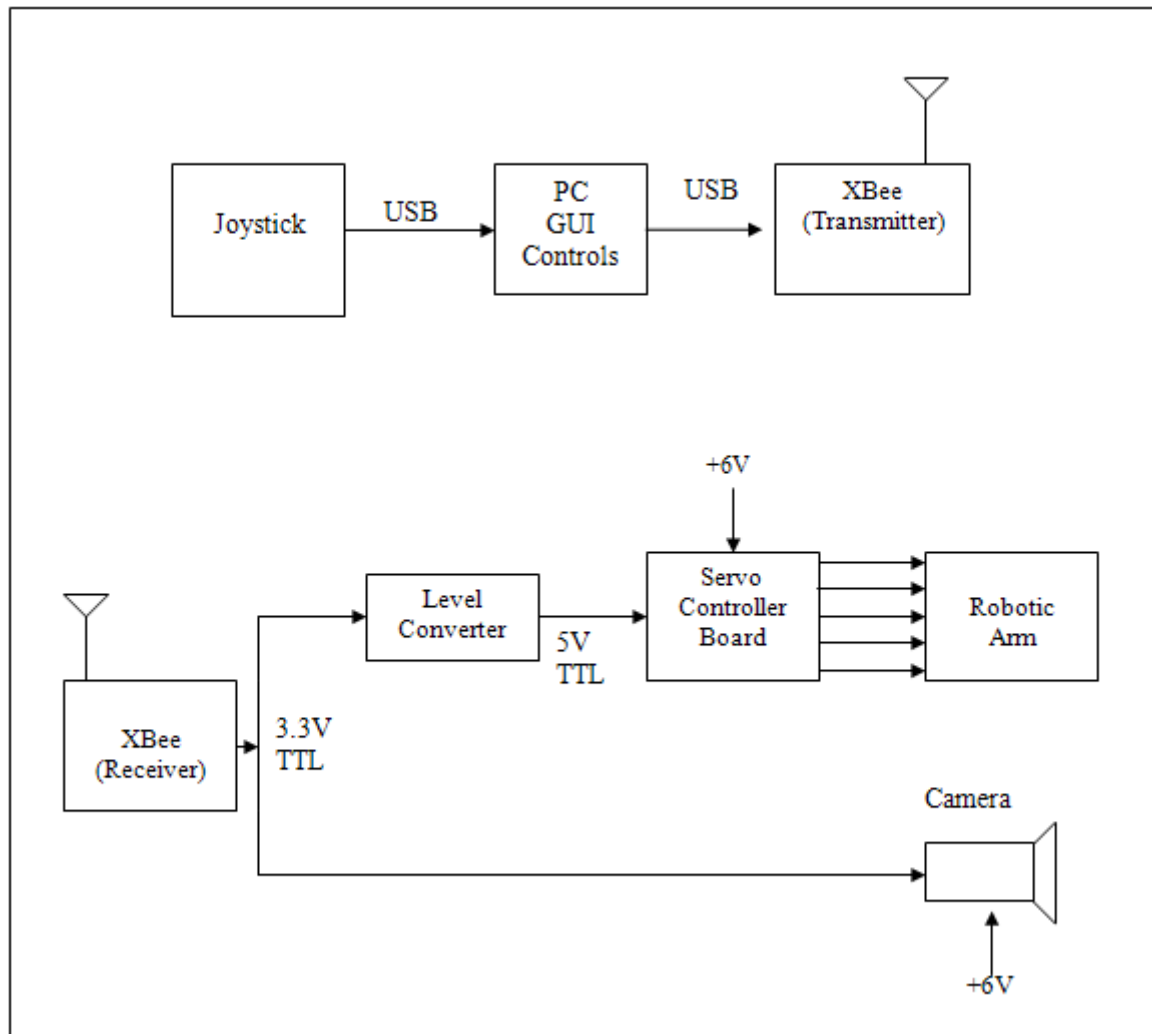
5V → V_{cc}

U0R → T_x

U0T → R_x

While uploading code in ESP32 CAM, connect GPIO to GND

2.2 BLOCK DIAGRAM



2.3 HARDWARE ANALYSIS

a) ESP32 CAM:

The ESP32-CAM is a very small camera module with the ESP32-S chip. Besides the OV2640 camera, and several GPIOs to connect peripherals, it also features a microSD card slot that can be useful to store images taken with the camera or to store files to serve to clients.

Features

- The smallest 802.11b/g/n Wi-Fi BT SoC module
- Low power 32-bit CPU, can also serve the application processor
- Up to 160MHz clock speed, summary computing power up to 600 DMIPS
- Built-in 520 KB SRAM, external 4MPSRAM
- Supports UART/SPI/I2C/PWM/ADC/DAC
- Support OV2640 and OV7670 cameras, built-in flash lamp
- Support image WiFi upload
- Support TF card
- Supports multiple sleep modes
- Embedded Lwip and FreeRTOS
- Supports STA/AP/STA+AP operation mode
- Support Smart Config/AirKiss technology
- Support for serial port local and remote firmware upgrades (FOTA)



Pin Details:

There are three GND pins and two pins for power: either 3.3V or 5V.

GPIO 1 and GPIO 3 are the serial pins. You need these pins to upload code to your board. Additionally, GPIO 0 also plays an important role, since it determines whether the ESP32 is in flashing mode or not. When GPIO 0 is connected to GND, the ESP32 is in flashing mode.

The following pins are internally connected to the microSD card reader:

GPIO 14: CLK

GPIO 15: CMD

GPIO 2: Data 0

GPIO 4: Data 1 (also connected to the on-board LED)

GPIO 12: Data 2

GPIO 13: Data 3

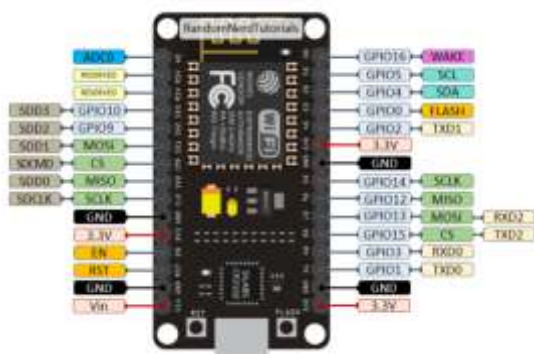
b)NodeMCU ESP8266:

NodeMCU is an open-source Lua based firmware and development board specially targeted for IoT based Applications. It includes firmware that runs on the ESP8266 Wi-Fi SoC from Espressif Systems, and hardware which is based on the ESP-12 module.

Features

- Microcontroller: Tensilica 32-bit RISC CPU Xtensa LX106
- Operating Voltage: 3.3V
- Input Voltage: 7-12V
- Digital I/O Pins (DIO): 16
- Analog Input Pins (ADC): 1
- UARTs: 1
- SPIs: 1
- I2Cs: 1
- Flash Memory: 4 MB
- SRAM: 64 KB
- Clock Speed: 80 MHz
- USB-TTL based on CP2102 is included onboard, Enabling Plug n Play
- PCB Antenna

Pin out:



Label	GPIO	Input	Output	Notes
D0	GPIO16	no interrupt	no PWM or I2C support	HIGH at boot used to wake up from deep sleep
D1	GPIO5	OK	OK	often used as SCL (I2C)
D2	GPIO4	OK	OK	often used as SDA (I2C)
D3	GPIO0	pulled up	OK	connected to FLASH button, boot fails if pulled LOW
D4	GPIO2	pulled up	OK	HIGH at boot connected to on-board LED, boot fails if pulled LOW
D5	GPIO14	OK	OK	SPI (SCLK)
D6	GPIO12	OK	OK	SPI (MISO)
D7	GPIO13	OK	OK	SPI (MOSI)
D8	GPIO15	pulled to GND	OK	SPI (CS) Boot fails if pulled HIGH
RX	GPIO3	OK	RX pin	HIGH at boot
TX	GPIO1	TX pin	OK	HIGH at boot debug output at boot, boot fails if pulled LOW
A0	ADC0	Analog Input	X	

c)MG90S Servo Motors

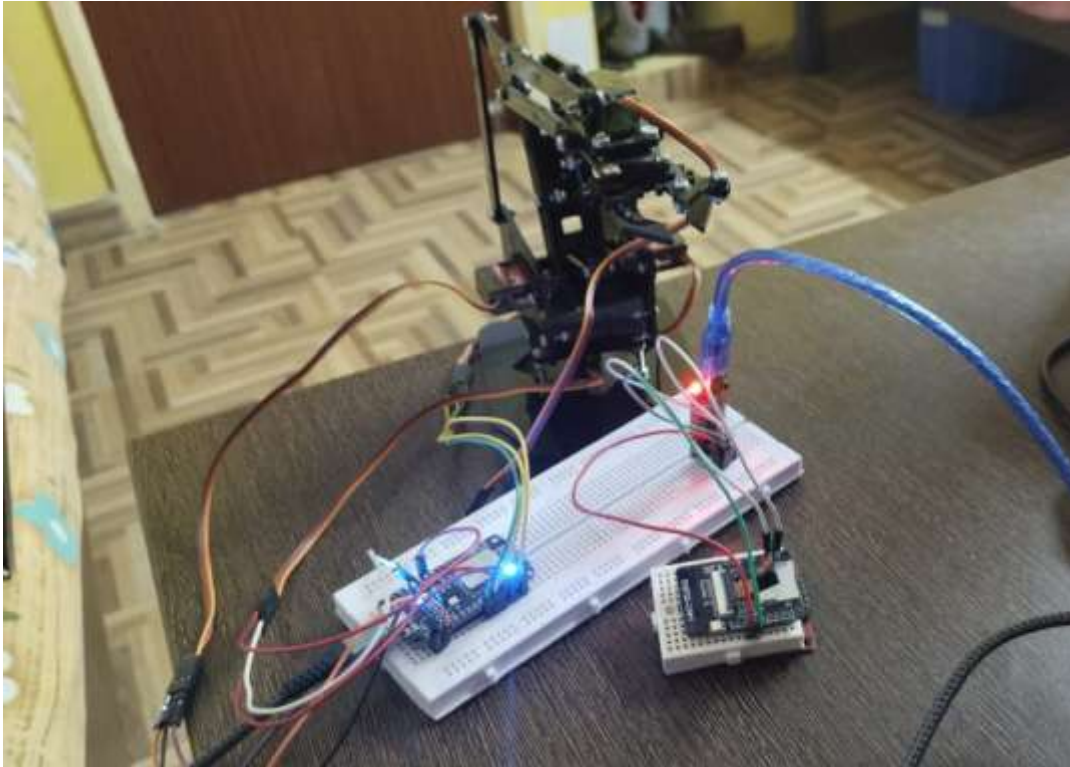
The MG90s servo motor is a compact and reliable motor designed for precise control in robotics and RC projects. With its metal gear construction and torque of up to 2.2 kg/cm, it offers powerful and accurate movement.

Features:

- High resolution
- Accurate positioning
- Fast control response
- Constant torque throughout the servo travel range
- Excellent holding power



2.4 HARDWARE SNAPSHOTS



2.4 CODE

Control Using Blynk Server

```
#define BLYNK_TEMPLATE_ID "TMPL3-gSgRLyq"
#define BLYNK_TEMPLATE_NAME "Robotic Arm"
#define BLYNK_AUTH_TOKEN
"UoHd_HR_YxXC78V0xLyq4fWA2TgtoJRN"

#define BLYNK_PRINT Serial
#include <ESP8266WiFi.h>

#include <BlynkSimpleEsp8266.h>
#include <Servo.h>

charauth[] = BLYNK_AUTH_TOKEN;
charssid[] = "wifi12";
char pass[] = "12345678";

Servo servo;
Servo servo1;
Servo servo2;
Servo servo3;

BLYNK_WRITE(V0){
servo.write(param.asInt());
}
BLYNK_WRITE(V1){
servo1.write(param.asInt());
}
BLYNK_WRITE(V2){
servo2.write(param.asInt());
}
BLYNK_WRITE(V3){
servo3.write(param.asInt());
}

void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);
```

```

Blynk.begin(auth,ssid,pass);
servo.attach(16);
servo1.attach(4);
servo2.attach(2);
servo3.attach(5);
}

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

```

ESP32 CAM

```

#include "esp_camera.h"
#include <WiFi.h>
#include "soc/soc.h"
#include "soc/rtc_cntl_reg.h"
//

// Select camera model
// #define CAMERA_MODEL_WROVER_KIT
// #define CAMERA_MODEL_M5STACK_PSRAM
#define CAMERA_MODEL_AI_THINKER

#include "camera_pins.h"

const char* ssid = "wifi12";
const char* password = "12345678";

void startCameraServer();

void setup() {
  WRITE_PERI_REG(RTC_CNTL_BROWN_OUT_REG, 0); //disable brownout
  detector
  Serial.begin(115200);
  Serial.setDebugOutput(true);
  Serial.println();

  camera_config_t config;

```

```

config.ledc_channel = LEDC_CHANNEL_0;
config.ledc_timer = LEDC_TIMER_0;
config.pin_d0 = Y2_GPIO_NUM;
config.pin_d1 = Y3_GPIO_NUM;
config.pin_d2 = Y4_GPIO_NUM;
config.pin_d3 = Y5_GPIO_NUM;
config.pin_d4 = Y6_GPIO_NUM;
config.pin_d5 = Y7_GPIO_NUM;
config.pin_d6 = Y8_GPIO_NUM;
config.pin_d7 = Y9_GPIO_NUM;
config.pin_xclk = XCLK_GPIO_NUM;
config.pin_pclk = PCLK_GPIO_NUM;
config.pin_vsync = VSYNC_GPIO_NUM;
config.pin_href = HREF_GPIO_NUM;
config.pin_sscb_sda = SIOD_GPIO_NUM;
config.pin_sscb_scl = SIOC_GPIO_NUM;
config.pin_pwdn = PWDN_GPIO_NUM;
config.pin_reset = RESET_GPIO_NUM;
config.xclk_freq_hz = 20000000;
config.pixel_format = PIXFORMAT_JPEG;
//init with high specs to pre-allocate larger buffers
if(psramFound()){
config.frame_size = FRAMESIZE_240X240;
config.jpeg_quality = 10;
config.fb_count = 2;
} else {
config.frame_size = FRAMESIZE_240X240;
config.jpeg_quality = 12;
config.fb_count = 1;
}

// camera init
esp_err_t err = esp_camera_init(&config);
if (err != ESP_OK) {
Serial.printf("Camera init failed with error 0x%x", err);
return;
}

//drop down frame size for higher initial frame rate
sensor_t * s = esp_camera_sensor_get();
s->set_framesize(s, FRAMESIZE_240X240);

WiFi.begin(ssid, password);

```



```

while (WiFi.status() != WL_CONNECTED) {
  delay(500);
  Serial.print(".");
}
Serial.println("");
Serial.println("WiFi connected");

startCameraServer();

Serial.print("Camera Ready! Use 'http://");
Serial.print(WiFi.localIP());
Serial.println("' to connect, the stream is on a different port channel 9601 ");
Serial.print("stream Ready! Use 'http://");
Serial.print(WiFi.localIP());
Serial.println(":9601/stream ");
Serial.print("image Ready! Use 'http://");
Serial.print(WiFi.localIP());
Serial.println("/capture ");

}

void loop() {
  // put your main code here, to run repeatedly:
  delay(10000);
}

```

Image Classification Header File

```

/* Edge Impulse ingestion SDK
 * Copyright (c) 2022 EdgeImpulse Inc.
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 * http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either
 * express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */

```

```

#ifndef _INFERENCE_H
#define _INFERENCE_H

// Undefine min/max macros as these conflict with C++ std min/max functions
// these are often included by Arduino cores
#include <Arduino.h>
#include <stdarg.h>
#ifdef min
#undef min
#endif // min
#ifdef max
#undef max
#endif // max
#ifdef round
#undef round
#endif // round
// Similar the ESP32 seems to define this, which is also used as an enum value in
TFLite
#ifdef DEFAULT
#undef DEFAULT
#endif // DEFAULT
// Infineon core defines this, conflicts with
CMSIS/DSP/Include/dsp/controller_functions.h
#ifdef A0
#undef A0
#endif // A0
#ifdef A1
#undef A1
#endif // A1
#ifdef A2
#undef A2
#endif // A2

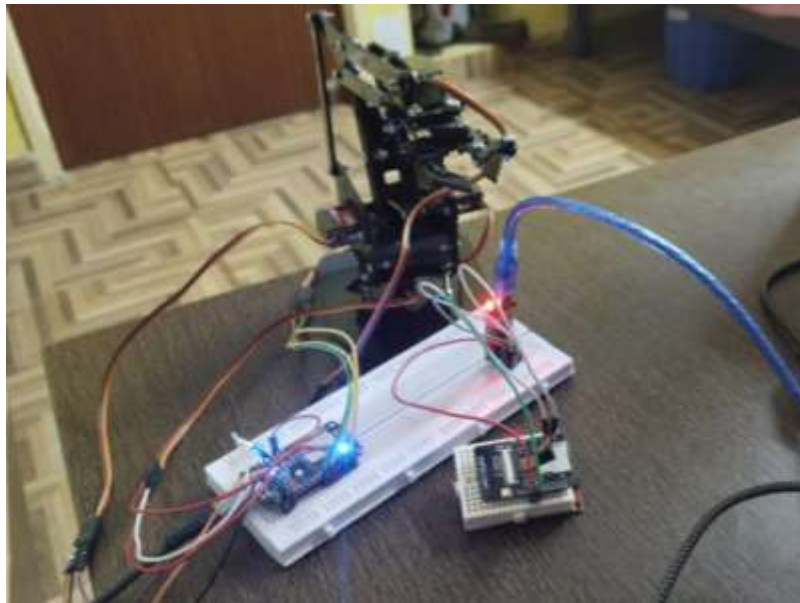
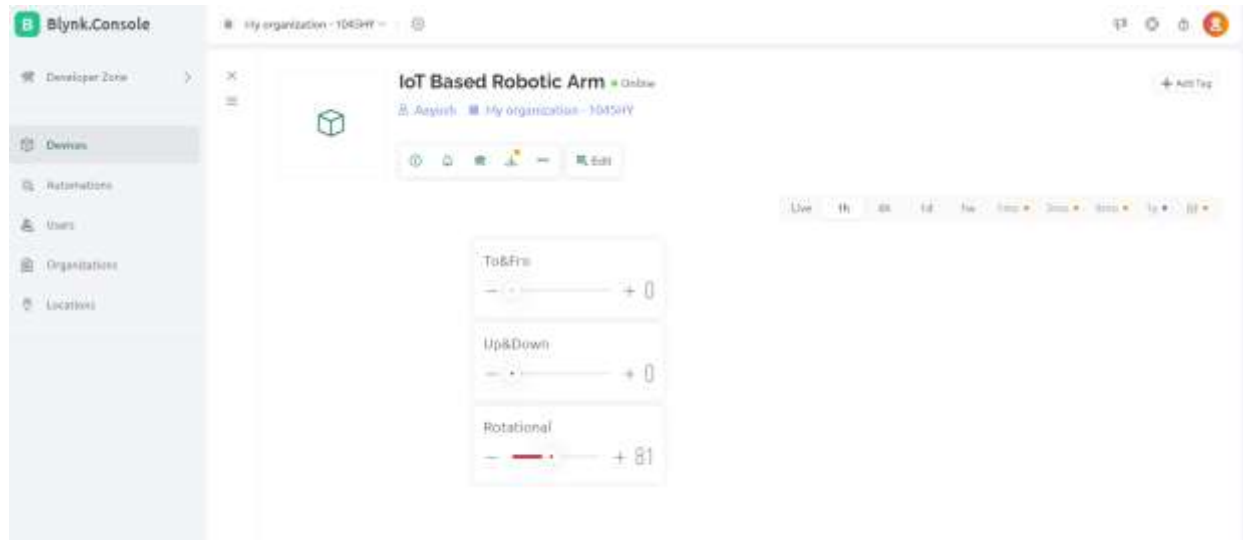
/* Includes ----- */
#include "edge-impulse-sdk/classifier/ei_run_classifier.h"
#include "edge-impulse-sdk/dsp/numpy.hpp"
#include "model-parameters/model_metadata.h"
#include "edge-impulse-sdk/classifier/ei_classifier_smooth.h"

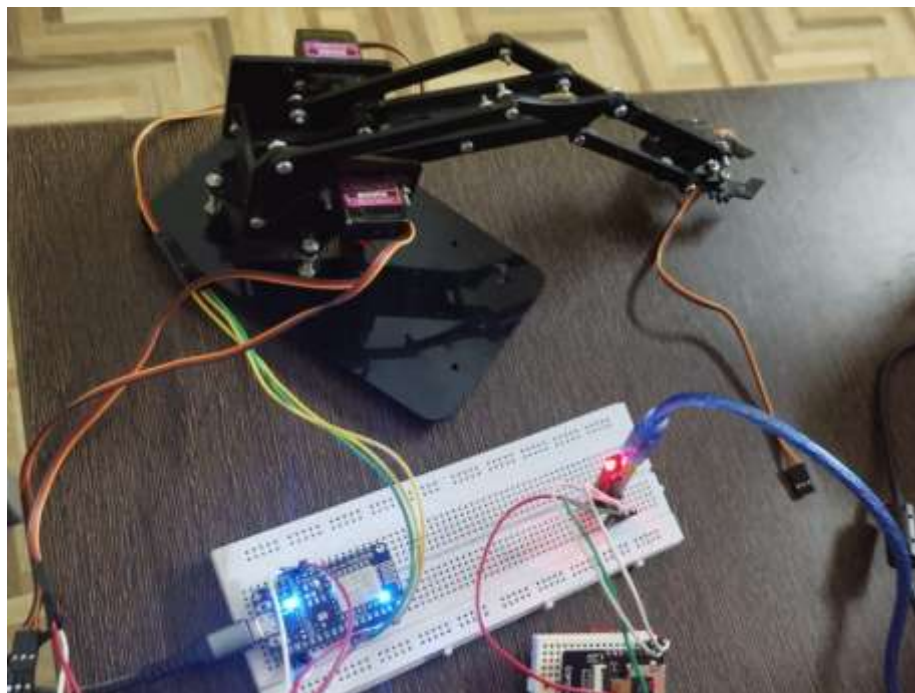
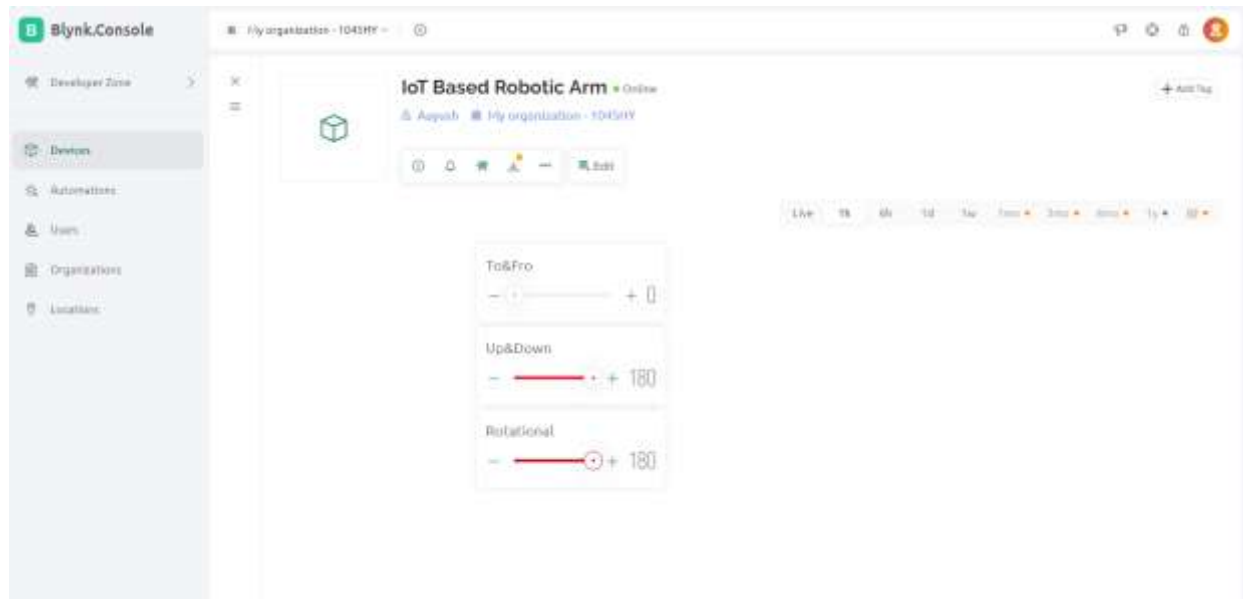
extern void ei_printf(const char *format, ...);

#endif // _INFERENCE_H

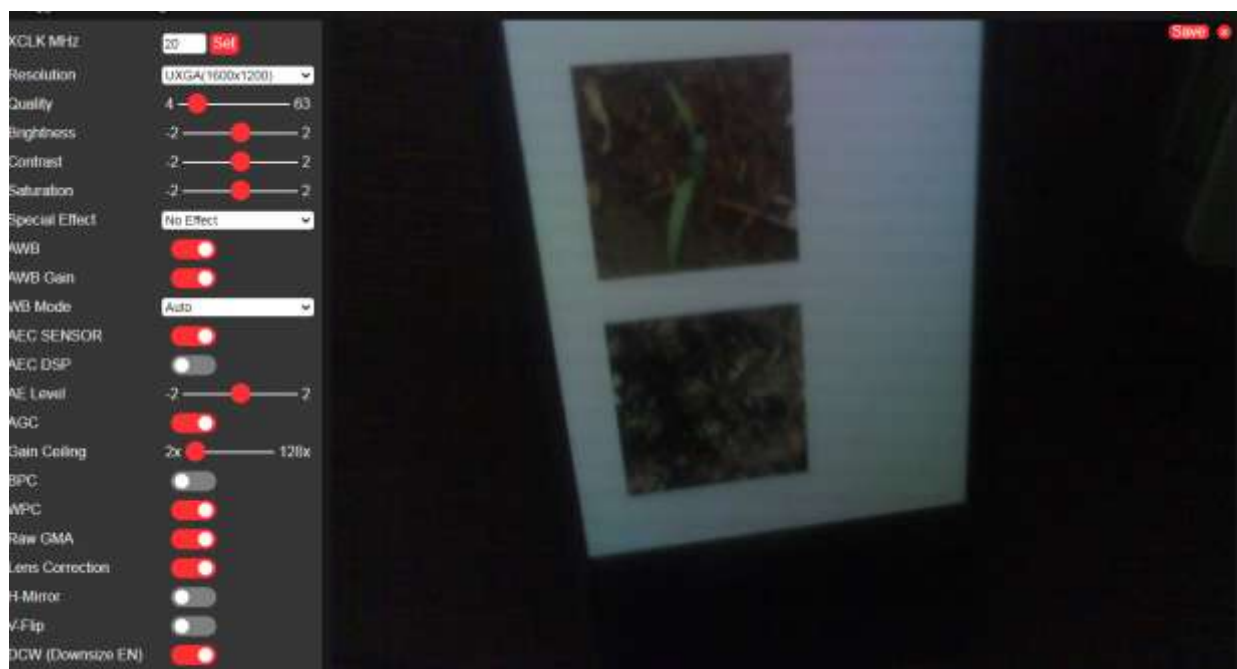
```

3. HARDWARE SCREENSHOT





ESP32 CAM



4.CONCLUSION AND RESULT

- **Successful Integration:** The completion of the smart weeding robotic arm project marks a successful integration of IoT and AI technologies for efficient weed management in agriculture.
- **Enhanced Efficiency:** Field tests have demonstrated the system's ability to accurately detect and remove weeds, leading to enhanced crop yield and reduced herbicide usage.
- **User-Friendly Interface:** The incorporation of the Blynk console has provided farmers with an intuitive interface for remote monitoring and manual control, simplifying operation and management.
- **Promising Outcomes:** The tangible outcomes of this project include a functional prototype that showcases the feasibility and effectiveness of the proposed solution, laying the groundwork for further advancements in precision agriculture.
- **Environmental Impact:** By minimizing herbicide usage and promoting sustainable farming practices, the smart weeding robotic arm has the potential to positively impact the environment and contribute to agricultural sustainability.

5.FUTURE WORK & LITRATURE SURVEY

- **AI Model Optimization:** Future work can focus on optimizing the AI models used for weed classification, drawing insights from recent literature on deep learning techniques for image recognition tasks (e.g., convolutional neural networks).
- **Sensor Integration:** Incorporating advanced sensing technologies such as multispectral imaging or LiDAR, as suggested in recent agricultural robotics research, can further improve the system's accuracy and reliability in weed detection.
- **Autonomous Navigation:** Exploration of autonomous navigation capabilities, inspired by recent studies on autonomous agricultural robots, can enable the robotic arm to navigate autonomously in complex field environments while performing weed management tasks.
- **Data Fusion Techniques:** Leveraging data fusion techniques, as discussed in the literature on precision agriculture, can enhance the system's decision-making capabilities by integrating data from multiple sources, such as imagery, weather conditions, and soil moisture levels.
- **Human-Robot Interaction:** Future research can explore human-robot interaction methodologies, including natural language processing and gesture recognition, to enhance user experience and enable more intuitive control mechanisms for farmers interacting with the robotic arm.

REFERENCES

1. Yogendra Sing Parihar ,(June 2019),“[IoT based Controlled Soilless vertical farmingwith hydroponics NFT system using microcontroller](#)“,”A review of use of Nodemcu ESP8266 in IoT products”.
2. [IsamIshaq&RashaAssaf](#), (December 2020), [Improving Irrigation by Using a CloudBased IoT System](#).
3. Sathya Narayanan, (December 2017), “A Cloud Based Irrigation System for Agriculture”.
4. NilsaMelo, (June 2020),” A Power Efficient IoT Edge Computing Solution for Cooking Oil Recycling”
5. [Alexandr D Lukyanov](#),(Jan 2021), “Estimation of the carbon footprint of IoT devices based on ESP8266 microcontrollers”

BIO DATA

Name:SATTWIK GHATAK

Mobile Number: 9854065414

E-mail:sattwik.ghatak2021@vitstudent.ac.in

Address: Kolkata, West Bengal

Name: ANUSMITA PANJI

Mobile Number: 9123884249

E-mail:anusmita.panji2021@vitstudent.ac.in

Address: Kolkata, West Bengal

Name: ANUSHA DAS

Mobile Number: 8240690546

E-mail:anusha.das2021@vitstudent.ac.in

Address: Kolkata, West Bengal

Name: AAYUSH PANWAR

Mobile Number: 9599385504

E-mail:aayush.panwar2021@vitstudent.ac.in

Address: Delhi

Drive Link of the project video:

https://drive.google.com/file/d/1Dtw4ugZCn6qD584pA_uYrAphzmEVVq5r/view?usp=sharing