BANGLADESH UNIVERSITY OF ENGINEERING AND TECHNOLOGY
DEPARTMENT OF ELECTRICAL AND ELECTRONIC ENGINEERING

EEE 318 (January 2023)

Control Systems Laboratory

Final Project Report

Section: B2 Group: 03

Autonomous Fire Extinguisher Bot

Course Instructors:

Shafin Bin Hamid
Md. Jawwad Ul Islam

Signature of Instructor: _

Academic Honesty Statement:

IMPORTANT! Please carefully read and sign the Academic Honesty Statement, below. Type the student ID and name, and put your signature. *You will not receive credit for this project experiment unless this statement is signed in the presence of your lab instructor.*

"In signing this statement, We hereby certify that the work on this project is our own and that we have not copied the work of any other students (past or present), and cited all relevant sources while completing this project. We understand that if we fail to honor this agreement, We will each receive a score of ZERO for this project and be subject to failure of this course."

Signature: _____ **Full Name: Shadman Sobhan** **Student ID: 1906109**	**Signature:** _____ **Full Name:Fahim Ahmed** **Student ID:1906110**
Signature: _____ **Full Name: Atiqur Rahaman Sohan** **Student ID:1906111**	**Signature:** _____ **Full Name: Kazi Abrar Mahmud** **Student ID:1906112**
Signature: _____ **Full Name: Md. Meraz Rahman** **Student ID:1906118**	**Signature:** _____ **Full Name: Md. Fahim** **Student ID:1206125**

Table of Contents

1 Abstract

In response to the increasing frequency and severity of wildfires, our project presents the design and implementation of a cutting-edge Image Processing-Based Fire-Bot. Leveraging advanced computer vision techniques, this autonomous system detects and responds to fire outbreaks in real-time. Through a combination of imaging and machine learning algorithms, the Fire-Bot swiftly identifies fire signatures, enabling rapid and precise intervention to mitigate potential disasters. This project not only showcases the power of technology in safeguarding lives and property but also serves as a testament to innovation in the realm of fire safety and disaster management.

2 Introduction

The development of an Image Processing-Based Fire-Bot addresses a complex engineering challenge due to the multifaceted nature of fire detection and response. It involves integrating various technologies such as machine learning, control system based robotics, and real-time data processing into a single cohesive system. The complexity arises from the need to accurately identify fires in diverse environmental conditions, distinguish them from false positives, and coordinate an autonomous response. Additionally, considerations like power management, mobility, and safety protocols further add to the intricacy of the problem. Solving this challenge requires a multidisciplinary approach, combining expertise in computer vision, robotics, and electrical engineering to create a reliable and efficient Fire-Bot system.
There are alternative methods and technologies that can be considered for fire detection and response beyond an Image Processing-Based Fire-Bot:

Smoke and Gas Sensors: Traditional smoke and gas sensors can provide early indications of a fire. These sensors can trigger alarms or automated fire suppression systems.

IoT and Sensor Networks: Deploying a network of IoT devices with environmental sensors (temperature, humidity, gas) can enable early detection through data analysis and anomaly detection.

Drones with Infrared Cameras: Drones equipped with infrared cameras can perform aerial surveillance and detect hotspots or flames in remote or inaccessible areas.

AI-Powered Data Analytics: Develop AI models that analyze historical and real-time data, including weather conditions, wind patterns, and terrain, to predict fire risk and enhance response planning.

The choice of method depends on factors like the specific use case, budget, environmental conditions, and the desired level of automation and accuracy in fire detection and response. A combination of these alternatives may also be considered for a comprehensive fire safety strategy.

3 Design

3.1 Problem Formulation

3.1.1 Identification of Scope

The scope of this project encompasses the design, development, and implementation of an Image Processing-Based Fire-Bot for early fire detection and response. It includes the integration of computer vision and machine learning technologies, thermal imaging sensors, and robotic mobility. The project aims to achieve real-time fire detection accuracy, autonomous navigation in diverse environments, and swift response capabilities. Additionally, considerations for power management, safety protocols, and system reliability are within scope. The project does not encompass full-scale firefighting but focuses on the early detection and monitoring of fire incidents.

3.1.2 Literature Review

The major scopes was to train a machine learning model using a micro-controller instead of a full-fledged SBC(single board computer).In order to achieve this we utilized an online utility called edge impulse, which compiles trained FOMO classifier into executable C library.This library can directly be incorporated with an micro-controller.The integration of robotics and autonomous navigation into fire detection systems has garnered attention. Studies have investigated the mobility and maneuverability of robots in complex terrains, enabling them to access remote areas and respond rapidly to fire outbreaks.For our project we opted for IMU based odometry ,The data form IMU is processed using a propitiatory fusion algorithm supplied by MotionApp corporation.

3.1.3 Formulation of Problem

The problem at hand involves designing and implementing an Image Processing-Based Fire-Bot, capable of early fire detection and swift autonomous response. The core challenges within this problem are Real-time-fire detection,autonomous navigation,power management and user interference.For real time fire detection we need to develop a Machine learning model and an stereoscopic camera system ,synchronizing the camera data a pre-defined data structure.For the case of navigation we need to develop a stable differential steering system.Here we need to apply a PID controller to incorporate the odometric data with the movement of a robot.In order successfully determine the controllers parameter we need to drive each motor with regulated voltage. So we also need a buck converter based voltage regulation within our rover body.

3.1.4 Analysis

This project combines advanced technologies such as computer vision, machine learning, thermal imaging, and robotics to create an Image Processing-Based Fire-Bot. It addresses the critical challenge of early fire detection and swift response in diverse environments. The project has the potential to significantly enhance fire safety and disaster management, mitigating the impact of wildfires and indoor fires.

3.2 Design Method

Image-processing based fire detection:

The hardware that we used is an esp32 camera,which is a micro-controller based imaging system.We used edge impulse utility to train a FOMO(Fast Object More Object) machine learning model. The training description is given below:

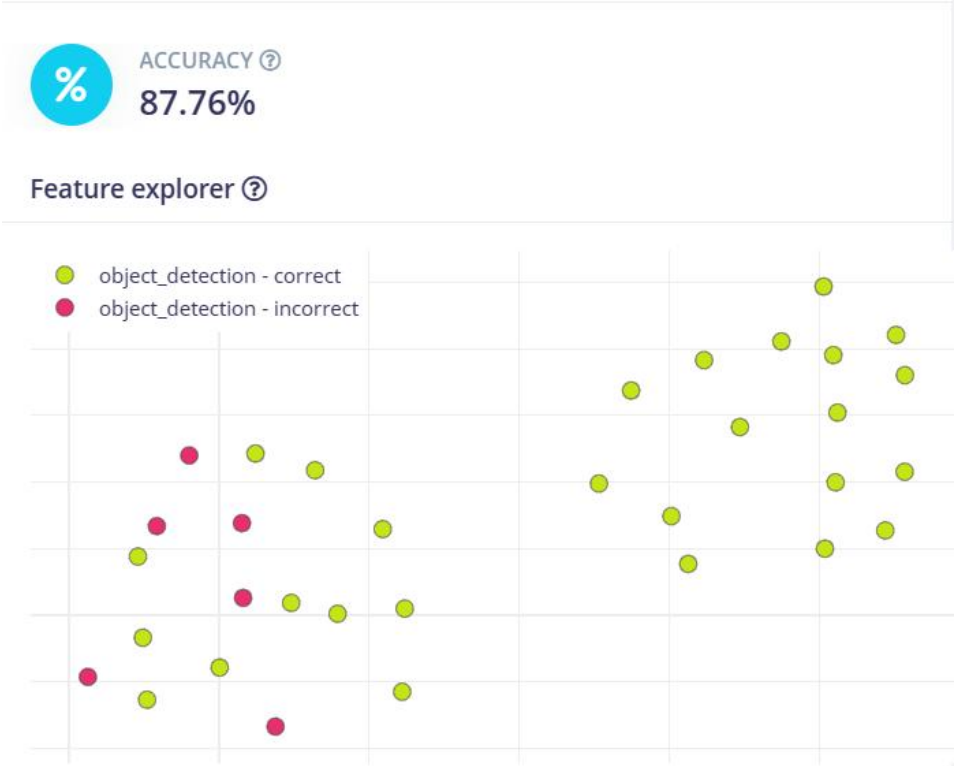


Fig : model testing result based on 20% partitioning scheme.

Here ,we can see that the model accuracy is 87.76% .hence there are some false detection cases within the partitioning.Now will put some training images through the classifier to demonstrate possible corner cases and comprehend some reason for false detection.

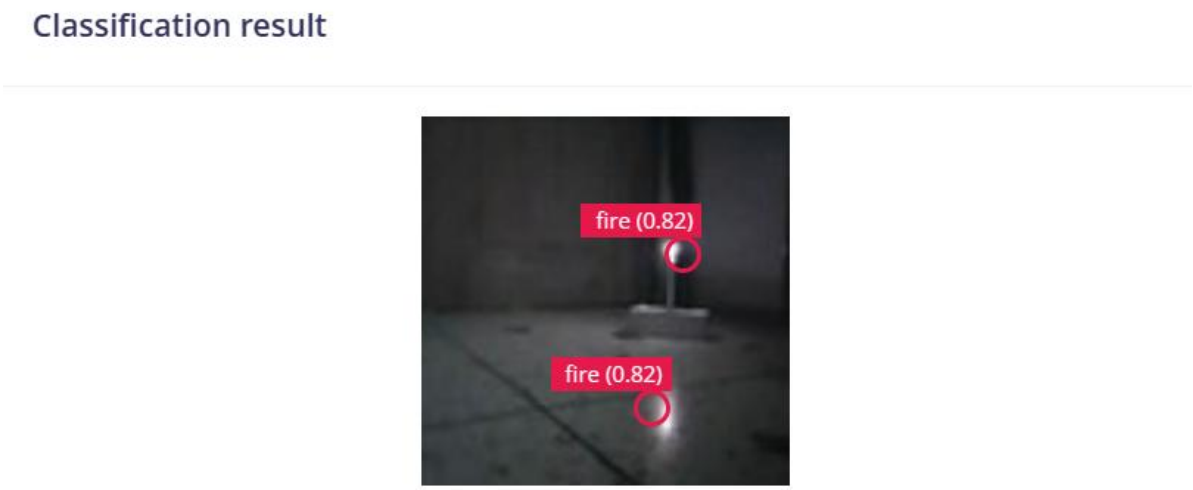


Fig : false classification result.

If we look closely ,we would see that the reflection of fire also classified as possible source of fire . It is quite difficult to mitigate such false detection using only imaging. One way this can be avoided is by fusing another sensor with the camera that can provide information of heat of each source. But we were prohibited to do so as part of a challenge of our project.So we tried our best to improve this model so that the accuracy would improve.

Few instances of our training data is given below:
Positive training image(Images that contain fire source)

Fig : Sample training data.(Positive images)
Negative training image(Images that doesn't contain any fire source fire source)

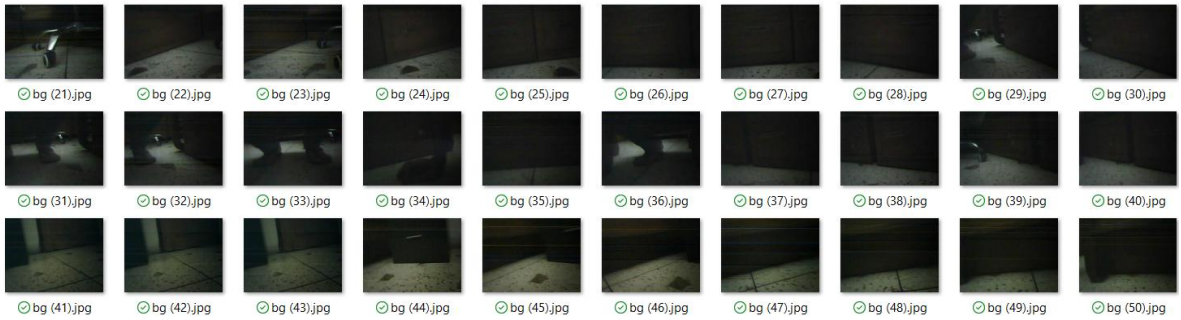


Fig : Sample training data.(Negative images)

In order to improve our detection rate we also introduced some hardware modification. We removed the IR filter from our esp32-camera module.This information allows more radiation to enter into the camera sensor and hence improve the inferencing.

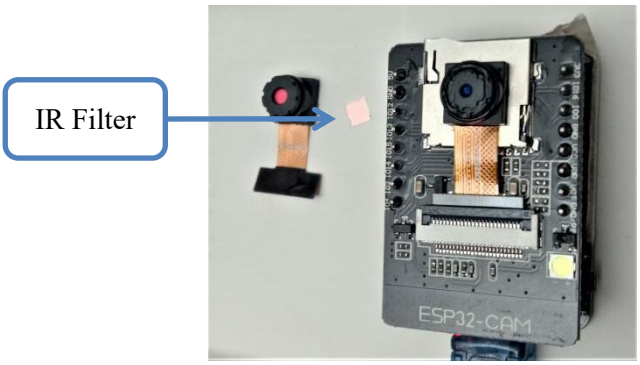


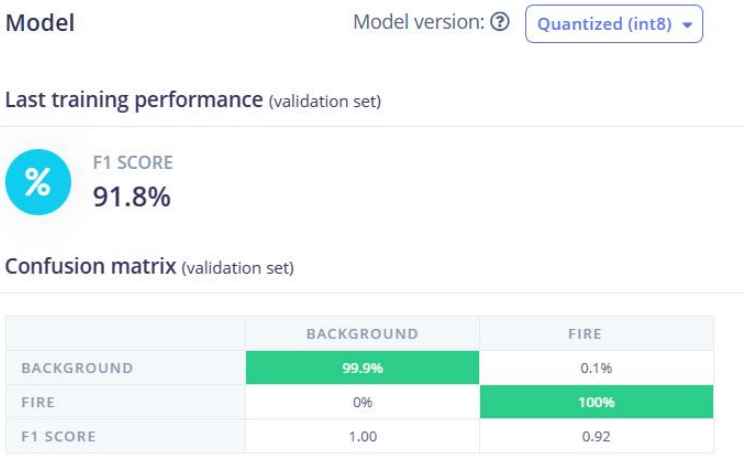
Fig: Removed IR filter from the camera module.

This modification improved our detection rate to approximately 92%. during the compilation of our code we configured the camera in such way that the exposer and brightness level of our sensor is low, hence only bright points form a fire source will be focused in the detection.
The code snippet for this procedure is given below:

```
390   //Buttons for disableing AEC sensor and AEC DSP
391   s->set_exposure_ctrl(s,0); //AEC sensor
392   s->set_aec2(s,0); //AEC DSP
393
394   //set value for AE Level and Exposer:
395   s->set_ae_level(s,-1);
396   s->set_aec_value(s,200);
397
398   s->set_brightness(s,-2);
```

Fig : Code snippet for image sensor configuration.

Finally the performance of the model is analyzed:



The figure above gives us FT score of our trained model which is 91.8%. The confusion matrix shows us cross validation accuracy of our model.Now we use edge impulse to convert this trained model into a executable C library. Edge impulse offered us an optimized (int8) variant model. Where detection time is reduced to 1612ms (from 2937ms) .For a 128kB RAM board which is running an ML model this inferencing speed is sufficient and probably the lowest that we can acquire from this board.

Quantized (int8)

Selected ✔

	IMAGE	OBJECT DETECTI...	TOTAL
LATENCY	15 ms.	1,612 ms.	**1,627 ms.**
RAM	4.0K	239.1K	**239.1K**
FLASH	-	72.7K	-
ACCURACY			-

Unoptimized (float32)

Select

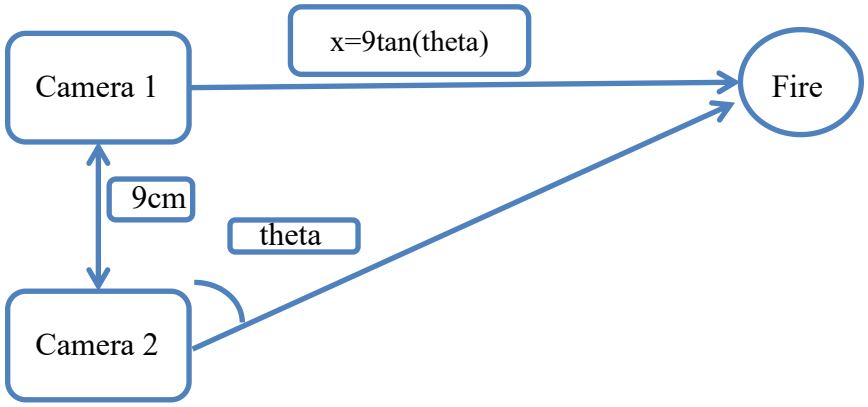
	IMAGE	OBJECT DETECTI...	TOTAL
LATENCY	15 ms.	2,937 ms.	**2,952 ms.**
RAM	4.0K	893.8K	**893.8K**
FLASH	-	95.8K	-
ACCURACY			-

Fig: Result of our optimized C library(edge impulse).

Distance measurements :

Inorder to infer distance we opted for stereoscopic camera setup. We used veroboard to mount both cameras within a per-defined distance.Then trigonometric formulation is used to evaluate the distance from fire source.This procedure is not the most accurate but gives us a somewhat approximate estimation of the distance.

We used the setup below:



Odometry:

Now in order to give our rover body a sense of its relative position ,we used MPU6050 IMU(Inertial Measurements Unit) based odometric system. The key feature of our system is that it can hold its angular position.The system acquires angular position from IMU and uses it as the input reference of a PID controller .The structure of PID controller that is used is given below.

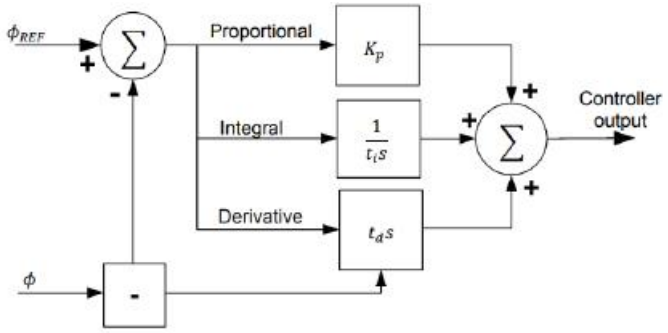


Fig: PID controller block diagram.

For better result we incorporated the output angular position with the differentiator branch.This loop is designed to hold a vehicles angular position relatively.but if we want to move our vehicle in forward direction we need to introduce offset in each motors .This offset is applied discretely when necessary.

To incorporate forward and backward motion we designed our controller through a ring structure.
Ring 1 - Holds angular position and updates every **10 micro** second.
Ring 2 - Introduces offset to forward motion and updates every **1micro** second.

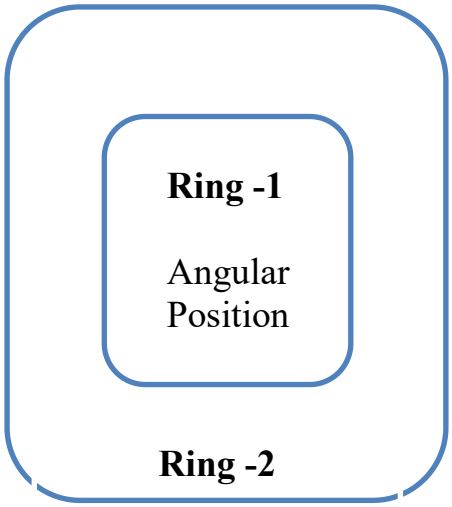
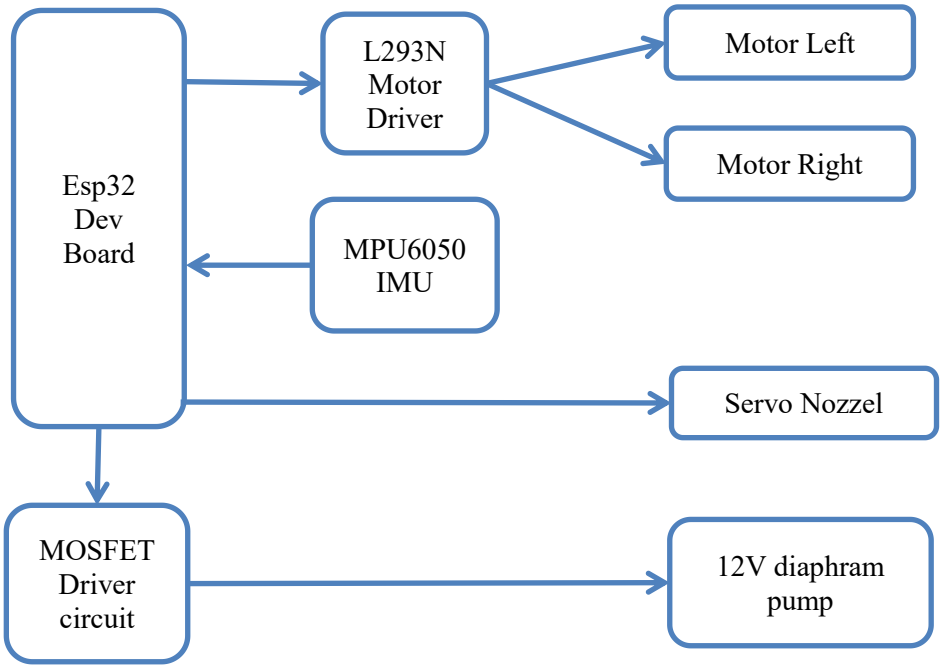


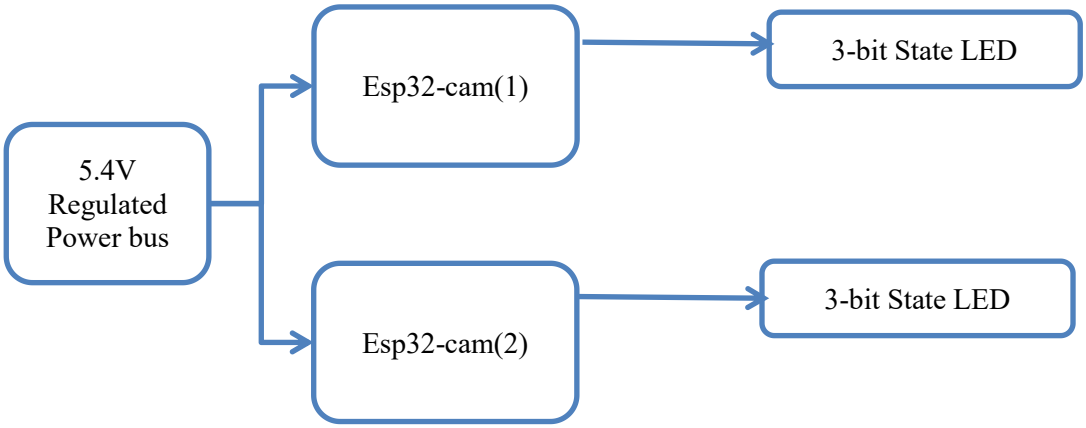
Fig : Hyper Structure of Rover motion control.

3.3 Circuit Diagram

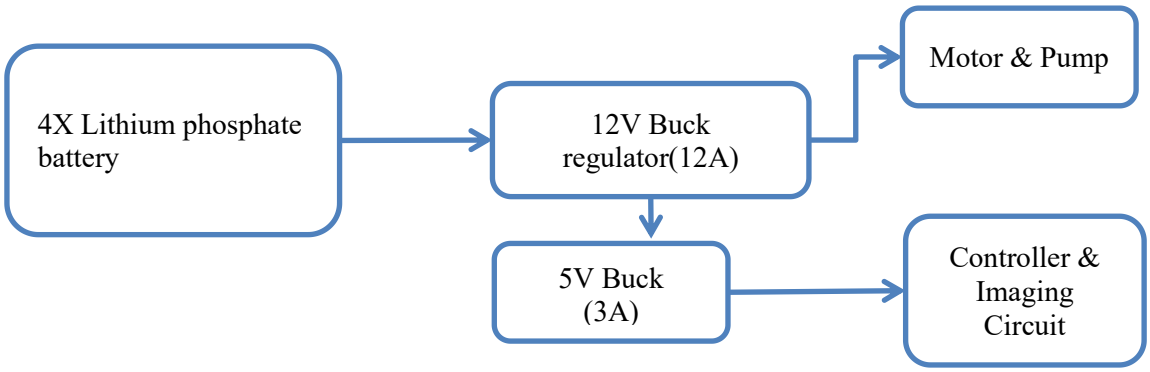
Here ,Block diagram of various circuit component is given below:
Main Controller Unit :



Stereoscopic Imaging unit:



Power Management:



3.4Full Source Code of Firmware

Source Code for **Main control unit**(esp32 dev board)

```
#include <Arduino.h>
#include "I2Cdev.h"
#include "pins_me.h"
#include <esp_now.h>
#include <WiFi.h>
#include "MPU6050_6Axis_MotionApps_V6_12.h"
#if I2CDEV_IMPLEMENTATION == I2CDEV_ARDUINO_WIRE
#include "Wire.h"
#endif



class SimplePID{
  private:
    float kp, kd, ki, umax; // Parameters
    float eprev, eintegral, valprev; // Storage

  public:
  // Constructor
  SimplePID() : kp(1), kd(0), ki(0), umax(255),
eprev(0.0), eintegral(0.0){}

  // A function to set the parameters
  void setParams(float kpIn, float kdIn, float kiIn,
float umaxIn){
    kp = kpIn; kd = kdIn; ki = kiIn; umax = umaxIn;
  }

  // A function to compute the control signal
  void evalu(float value,int val_sign, float
target,int trg_sign, float deltaT, int &pwr, int
&dir){
    // error
    float e = trg_sign*target + val_sign*value;

    // derivative
    float dedt = (e-eprev)/(deltaT);
    float dvaldt = val_sign*(value-valprev)/(deltaT);

    // integral
    eintegral = eintegral + e*deltaT;

    // control signal
    //float u = kp*e + kd*dedt + ki*eintegral;
    float u = kp*e + kd*(dvaldt+dedt) + ki*eintegral;

    // motor power
    pwr = (int) fabs(u); //Absolute value of float;
    if( pwr > umax ){
      pwr = umax;
    }

    // motor direction
    dir = 1;
    if(u<0){
      dir = -1;
    }

    // store previous error
    eprev = e;
    valprev = value;
  }

};
// Define a data structure
```

```
typedef struct struct_message {
  long cam_info[2] ; // camera_id ,camera_frame
  bool fire_avail;
  float predic_score;
  int x_;
  int y_;
  int width_;
  int height_;

} struct_message;
// Create a structured object
struct_message myData;


// Setting PWM properties
const int freq = 300000;
const int freq_servo = 50;
const int c1 = 0;
const int c2 = 1;
const int c3 = 2;
const int resolution = 8;
int dutyCycle = 200;

// Globals
long t_prev=0;
volatile int pos_l =0;
volatile int pos_r =0;

//float target[16]
={0,22.5,45,67.5,90,112.5,135,157.5,180,202.5,225,247.
5,270,292.5,315,337.5};
float target[15] ={-157.5,-135,-112.5,-90,-67.5,-45,-
22.5,0,22.5,45,67.5,90,112.5,135,157.5};
//float target[8] ={0,45,90,135,180,225,270,315};
int buff = 7; // starts at 0 degree
int sgn = 1;
float align_angle;
int speed = 0;
volatile long t_prev_speed =
0,t_prev_pump=0,currT,currT_speed;
//long currT


float tmp_angl = 0;
bool found1 = false;// detection for first camera
int frame1 = 0;
int score1 = 0;//how many of 5 frames contain fire

bool found2 = false;// detection for second camera
int frame2 = 0;
int score2 = 0;//how many of 5 frames contain fire
float dist = 0;
long time_req = 0;
int itr = 0;
bool reached = false;
int pump_cycle = 0;

bool dmpReady = false;  // set true if DMP init was
successful
volatile bool mpuInterrupt = false;     // indicates
whether MPU interrupt pin has gone high
uint8_t mpuIntStatus;   // holds actual interrupt
status byte from MPU
uint8_t devStatus;      // return status after each
device operation (0 = success, !0 = error)
uint16_t packetSize;    // expected DMP packet size
(default is 42 bytes)
uint16_t fifoCount;     // count of all bytes
currently in FIFO
uint8_t fifoBuffer[64]; // FIFO storage buffer

// orientation/motion vars
Quaternion q;           // [w, x, y,
z]         quaternion container
VectorInt16 aa;         // [x, y, z]            accel
sensor measurements
VectorInt16 gy;         // [x, y, z]            gyro
sensor measurements
VectorInt16 aaReal;     // [x, y,
z]           gravity-free accel sensor measurements
VectorInt16 aaWorld;    // [x, y, z]            world-
frame accel sensor measurements
VectorFloat gravity;    // [x, y,
z]           gravity vector
float euler[3];         // [psi, theta, phi]    Euler
angle container
float ypr[3]={0.0, 0.0, 0.0},ypr_prev[3]={0.0, 0.0,
0.0};           // [yaw, pitch, roll]   yaw/pitch/roll
container and gravity vector


// PID class instances
SimplePID pid_left;
SimplePID pid_right;
MPU6050 mpu;

void IRAM_ATTR isr_right() {
  int c = digitalRead(enc_right_y);
  if(c>0)
  {pos_r--;}
  else
  {pos_r++;}
}

void IRAM_ATTR isr_left() {
  int b = digitalRead(enc_left_y);
  if(b>0)
  {pos_l--;}
  else
  {pos_l++;}
}
void dmpDataReady() {
  mpuInterrupt = true;
}
// Callback function executed when data is received
void OnDataRecv(const uint8_t * mac, const uint8_t
*incomingData, int len) {
  memcpy(&myData, incomingData, sizeof(myData));
}

void setup() {
   // join I2C bus (I2Cdev library doesn't do this
automatically)
  #if I2CDEV_IMPLEMENTATION == I2CDEV_ARDUINO_WIRE
    Wire.begin();
```

```
    Wire.setClock(400000); // 400kHz I2C clock.
Comment this line if having compilation difficulties
  #elif I2CDEV_IMPLEMENTATION ==
I2CDEV_BUILTIN_FASTWIRE
    Fastwire::setup(400, true);
  #endif
  Serial.begin(115200);


  // Set ESP32 as a Wi-Fi Station
  WiFi.mode(WIFI_STA);
    // Initilize ESP-NOW
  if (esp_now_init() != ESP_OK) {
    Serial.println("Error initializing ESP-NOW");
    return;
  }

  // Register callback function
  esp_now_register_recv_cb(OnDataRecv);


  pinMode(servo_pin,OUTPUT);
  pinMode(pump_pin,OUTPUT);

  pinMode(IN_1, OUTPUT);
  pinMode(IN_2, OUTPUT);
  pinMode(EN_left, OUTPUT);
  pinMode(enc_left_y, INPUT);
  pinMode(enc_left_g, INPUT);
  attachInterrupt(enc_left_g, isr_left, RISING);
  pid_left.setParams(2.9,0.03,1/120000,90);
//55,0.75,0,25 //1.75,0.42,0.0001,75// new
sys :5.75,3.42,0.00001,95//20/3,80/3,0.00001,95

  pinMode(IN_3, OUTPUT);
  pinMode(IN_4, OUTPUT);
  pinMode(EN_right, OUTPUT);
  pinMode(enc_right_y, INPUT);
  pinMode(enc_right_g, INPUT);
  attachInterrupt(enc_right_g, isr_right, RISING);
  pid_right.setParams(2.9,0.03,1/120000,95); //best
1,0,0,150


  // configure LED PWM functionalitites
  ledcSetup(c1, freq, resolution);
  ledcSetup(c2, freq, resolution);
  ledcSetup(c3, freq_servo , resolution);
  // attach the channel to the GPIO to be controlled
  ledcAttachPin(EN_left, c1);
  ledcAttachPin(EN_right, c2);
  ledcAttachPin(servo_pin, c3);


  delay(2000);
    // initialize device
  Serial.println(F("Initializing I2C devices..."));
  mpu.initialize();
  pinMode(GYRO_INTERRUPT_PIN, INPUT);

  // verify connection
  Serial.println(F("Testing device connections..."));
  Serial.println(mpu.testConnection() ? F("MPU6050
connection successful") : F("MPU6050 connection
failed"));

  delay(200);
  // load and configure the DMP
  Serial.println(F("Initializing DMP..."));
  devStatus = mpu.dmpInitialize();

  // supply your own gyro offsets here, scaled for
min sensitivity
  mpu.setXGyroOffset(421);
  mpu.setYGyroOffset(-50);
  mpu.setZGyroOffset(214);
  mpu.setXAccelOffset(-4809);
  mpu.setYAccelOffset(-763);
  mpu.setZAccelOffset(1855);
  // make sure it worked (returns 0 if so)
  if (devStatus == 0) {
    // Calibration Time: generate offsets and
calibrate our MPU6050
    mpu.CalibrateAccel(6);
    mpu.CalibrateGyro(6);
    Serial.println();
    mpu.PrintActiveOffsets();
    // turn on the DMP, now that it's ready
    Serial.println(F("Enabling DMP..."));
    mpu.setDMPEnabled(true);

    // enable Arduino interrupt detection
    Serial.print(F("Enabling interrupt detection
(Arduino external interrupt "));
    Serial.print(digitalPinToInterrupt(GYRO_INTERRUPT
_PIN));
    Serial.println(F(")..."));
    attachInterrupt(digitalPinToInterrupt(GYRO_INTERR
UPT_PIN), dmpDataReady, RISING);
    mpuIntStatus = mpu.getIntStatus();

    // set our DMP Ready flag so the main loop()
function knows it's okay to use it
    Serial.println(F("DMP ready! Waiting for first
interrupt..."));
    dmpReady = true;

    // get expected DMP packet size for later
comparison
    packetSize = mpu.dmpGetFIFOPacketSize();
  } else {
    // ERROR!
    // 1 = initial memory load failed
    // 2 = DMP configuration updates failed
    // (if it's going to break, usually the code will
be 1)
    Serial.print(F("DMP Initialization failed (code
"));
    Serial.print(devStatus);
    Serial.println(F(")"));
  }

}

void loop() {
```

```
  currT = micros();

    if(!found1 && itr>15500) // if detected stop
angle increment and align
    {
      itr = 0;
      buff = buff+sgn*1;
      if(buff>=14 || buff<=0)
        sgn = -1*sgn;
    }
  if((myData.cam_info[0] == 1)&&!found1)
  {
    if(myData.fire_avail)
      {
        float val  = myData.x_;
         tmp_angl =  tmp_angl + map(val,0,88,-20,20);
         score1++;
      }
    frame1 = myData.cam_info[1];
  }

  if(frame1 >=7)
  {
    frame1 = 0;
    if(score1>=5)
    {
      align_angle = tmp_angl/score1;
      found1 = true;
      score1 = 0;
      tmp_angl = 0;
      t_prev=currT;

    }
  }
//.............For Camera
2.....................................

  if((myData.cam_info[0] == 2)&&!found2&&((currT-
t_prev)>5e6)&&(frame2<myData.cam_info[1]))
  {
    if(myData.fire_avail)
      {
        float val  = myData.x_;
         tmp_angl =  tmp_angl + map(val,0,88,-20,20);
         score2++;
      }
    //frame2++;
    frame2 = myData.cam_info[1];
  }

  if(frame2 >=20 && !found2)
  {
    frame2 = 0;
    if(score2>=18)
    {
      tmp_angl = 90-tmp_angl/score2;
      // Required distance calculation
      dist = 9*tan(tmp_angl*M_PI/180);

      //Serial.println(tmp_angl);
      time_req = (dist/24.56)*1e6; // in micro
seconds
      Serial.println(dist/24.56);
      Serial.println(dist);
      currT_speed = micros();
      t_prev_speed=currT_speed; // initialize timer
      found2 = true;
      score2 = 0;
      tmp_angl = 0;

    }
  }
  if(found2&&!reached)
  {

    speed = 17;
    //Serial.println(currT-t_prev_speed);
    if((currT-t_prev_speed)>=0.97*(time_req))
    {
      speed = 0;
      t_prev_pump = currT;
      found2 = false;
      reached = true;
    }

  }

  if(reached && (pump_cycle<3))
  { // 22 -------> left side
    // 14 -------> right side
    digitalWrite(pump_pin,HIGH);
    if((currT-t_prev_pump)>=0.8e6)
      {
        ledcWrite(c3,14);
        //t_prev_pump = currT;
        if((currT-t_prev_pump)>=2*0.8e6)
        {
          t_prev_pump = currT;
          pump_cycle++;
        }
        //t_prev_pump = ((currT-
t_prev_pump)>=2*0.8e6)?currT:t_prev_pump;

      }else
        ledcWrite(c3,22);

  }
  else digitalWrite(pump_pin,LOW);


  //target_motion(currT,22+align_angle,speed);
  target_motion(currT,target[buff]+align_angle,speed);
  itr++;
  //Serial.println(ypr[0] * 180 / M_PI);


  //Serial.println(target[buff]+align_angle);
    // Serial.print(target[2]);
    // Serial.print(" ");
    // Serial.print(ypr[0] * 180 / M_PI);
    // Serial.println(" ");
  //Serial.println(buff);

}

void target_motion(float currT,float target,int speed)
```

```
{
  static long prevT = 0;
  static long prev_speedT = 0;
  float deltaT = ((float) (currT - prevT))/( 1.0e6 );


  if (mpu.dmpGetCurrentFIFOPacket(fifoBuffer)) { //
Get the Latest packet
  mpu.dmpGetQuaternion(&q, fifoBuffer);
  mpu.dmpGetGravity(&gravity, &q);
  mpu.dmpGetYawPitchRoll(ypr, &q, &gravity);
  }
  else
  {
    ypr[0]=ypr_prev[0];ypr[1]=ypr_prev[1];ypr[2]=ypr_
prev[2];
  }

  // Read the position
  float pos_l_tmp,pos_r_tmp;

  pos_l_tmp = ypr[0] * 180 / M_PI;
  //pos_l_tmp =
(pos_l_tmp<0)?(pos_l_tmp+360):(pos_l_tmp);
  pos_r_tmp = pos_l_tmp;
  //Serial.println(pos_l_tmp);
  int pwr_l, dir_l;
    // evaluate the control signal
  pid_left.evalu(pos_l_tmp,-
1,target,1,deltaT,pwr_l,dir_l);
  int pwr_r, dir_r;
    // evaluate the control signal
  pid_right.evalu(pos_r_tmp,1,target,-
1,deltaT,pwr_r,dir_r);
  int flg =1;
    // signal the motor
  if((currT-prev_speedT)>(1*10e-6)){
    pwr_l = constrain(pwr_l+dir_l*speed,0,255);
    pwr_r = constrain(pwr_r+dir_r*speed,0,255);
    setMotor(dir_l,pwr_l,c1,IN_1,IN_2);
    setMotor(dir_r,pwr_r,c2,IN_3,IN_4);
    prev_speedT = currT;
    flg = 0;
  }


  if(((currT-prevT)>(10e-5))&&flg){
    setMotor(dir_l,pwr_l,c1,IN_1,IN_2);
    setMotor(dir_r,pwr_r,c2,IN_3,IN_4);
    prevT = currT;
    flg=1;
  }

    ypr_prev[0] = ypr[0];
    ypr_prev[1] = ypr[1];
    ypr_prev[2] = ypr[2];
    //Serial.println(pos_l_tmp);
    //prevT = currT;

}

void setMotor(int dir,int pwmVal, int enx ,int
in1,int in2)
{
  ledcWrite(enx, pwmVal);
  if(dir == -1)
  {
      digitalWrite(in1,HIGH);
      digitalWrite(in2,LOW);

  }
  else if(dir == 1)
  {
      digitalWrite(in1,LOW);
      digitalWrite(in2,HIGH);

  }
  else
  {
      digitalWrite(in1,LOW);
      digitalWrite(in2,LOW);
  }
}
```

***Table**: Source Code for the main program*

Source Code for Each **camera module**

```
#include <fire_infrared_two_inferencing.h>
#include "edge-impulse-sdk/dsp/image/image.hpp"

#include "esp_camera.h"
#include <esp_now.h>
#include <WiFi.h>

//Pins
const int fire_avail_pin = 13; // Blue
const int esp_now_avail_pin = 14; // Red
const int initi_avail_pin = 4; // Green


// Select camera model - find more camera models in
camera_pins.h file here
// https://github.com/espressif/arduino-
esp32/blob/master/libraries/ESP32/examples/Camera/Cam
eraWebServer/camera_pins.h

//#define CAMERA_MODEL_ESP_EYE // Has PSRAM
#define CAMERA_MODEL_AI_THINKER // Has PSRAM

#if defined(CAMERA_MODEL_ESP_EYE)
```

```
#define PWDN_GPIO_NUM    -1
#define RESET_GPIO_NUM   -1
#define XCLK_GPIO_NUM    4
#define SIOD_GPIO_NUM    18
#define SIOC_GPIO_NUM    23

#define Y9_GPIO_NUM      36
#define Y8_GPIO_NUM      37
#define Y7_GPIO_NUM      38
#define Y6_GPIO_NUM      39
#define Y5_GPIO_NUM      35
#define Y4_GPIO_NUM      14
#define Y3_GPIO_NUM      13
#define Y2_GPIO_NUM      34
#define VSYNC_GPIO_NUM   5
#define HREF_GPIO_NUM    27
#define PCLK_GPIO_NUM    25

#elif defined(CAMERA_MODEL_AI_THINKER)
#define PWDN_GPIO_NUM     32
#define RESET_GPIO_NUM    -1
#define XCLK_GPIO_NUM      0
#define SIOD_GPIO_NUM     26
#define SIOC_GPIO_NUM     27

#define Y9_GPIO_NUM       35
#define Y8_GPIO_NUM       34
#define Y7_GPIO_NUM       39
#define Y6_GPIO_NUM       36
#define Y5_GPIO_NUM       21
#define Y4_GPIO_NUM       19
#define Y3_GPIO_NUM       18
#define Y2_GPIO_NUM        5
#define VSYNC_GPIO_NUM    25
#define HREF_GPIO_NUM     23
#define PCLK_GPIO_NUM     22

#else
#error "Camera model not selected"
#endif

/* Constant defines -------------------------------
----------------------- */
#define EI_CAMERA_RAW_FRAME_BUFFER_COLS           320
#define EI_CAMERA_RAW_FRAME_BUFFER_ROWS           240
#define EI_CAMERA_FRAME_BYTE_SIZE                 3

/* Private variables -------------------------------
----------------------- */
static bool debug_nn = false; // Set this to true to
see e.g. features generated from the raw signal
static bool is_initialised = false;
uint8_t *snapshot_buf; //points to the output of the
capture

static camera_config_t camera_config = {
    .pin_pwdn = PWDN_GPIO_NUM,
    .pin_reset = RESET_GPIO_NUM,
    .pin_xclk = XCLK_GPIO_NUM,
    .pin_sscb_sda = SIOD_GPIO_NUM,
    .pin_sscb_scl = SIOC_GPIO_NUM,

    .pin_d7 = Y9_GPIO_NUM,
    .pin_d6 = Y8_GPIO_NUM,
    .pin_d5 = Y7_GPIO_NUM,
    .pin_d4 = Y6_GPIO_NUM,
    .pin_d3 = Y5_GPIO_NUM,
    .pin_d2 = Y4_GPIO_NUM,
    .pin_d1 = Y3_GPIO_NUM,
    .pin_d0 = Y2_GPIO_NUM,
    .pin_vsync = VSYNC_GPIO_NUM,
    .pin_href = HREF_GPIO_NUM,
    .pin_pclk = PCLK_GPIO_NUM,

    //XCLK 20MHz or 10MHz for OV2640 double FPS
(Experimental)
    .xclk_freq_hz = 20000000,
    .ledc_timer = LEDC_TIMER_0,
    .ledc_channel = LEDC_CHANNEL_0,

    .pixel_format = PIXFORMAT_JPEG,
//YUV422,GRAYSCALE,RGB565,JPEG
    .frame_size = FRAMESIZE_QVGA,    //QQVGA-UXGA Do
not use sizes above QVGA when not JPEG

    .jpeg_quality = 12, //0-63 lower number means
higher quality
    .fb_count = 1,       //if more than one, i2s runs
in continuous mode. Use only with JPEG
    .fb_location = CAMERA_FB_IN_PSRAM,
    .grab_mode = CAMERA_GRAB_WHEN_EMPTY,
};

//.................................................
.................................................
//
//ESPNow
stuff.............................................
.........................................//

// MAC Address of responder - edit as required
uint8_t broadcastAddress[] = {0xB8, 0xD6, 0x1A, 0x43,
0x32, 0xD8}; //B8:D6:1A:43:32:D8


// Define a data structure
long cam_id = 1;
long frame = 0;
typedef struct struct_message {
  long cam_info[2] ; // camera_id ,camera_frame
  bool fire_avail;
  float predic_score;
  int x_;
  int y_;
  int width_;
  int height_;

} struct_message;

// Create a structured object
struct_message myData;

// Peer info
esp_now_peer_info_t peerInfo;

// Callback function called when data is sent
void OnDataSent(const uint8_t *mac_addr,
esp_now_send_status_t status) {
```

```
  // Serial.print("\r\nLast Packet Send Status:\t");
  // Serial.println(status == ESP_NOW_SEND_SUCCESS ? "Delivery Success" : "Delivery Fail");
  if(status != ESP_NOW_SEND_SUCCESS)
  {
    digitalWrite(esp_now_avail_pin, HIGH);
    Serial.println("Delivery Success");
  }

  else
  {
    digitalWrite(esp_now_avail_pin, LOW);
    Serial.println("Delivery Fail");
  }

}
/* Function definitions ------------------------------------------------------- */

bool ei_camera_init(void);
void ei_camera_deinit(void);
bool ei_camera_capture(uint32_t img_width, uint32_t img_height, uint8_t *out_buf) ;

/**
* @brief      Arduino setup function
*/
void setup()
{
    // put your setup code here, to run once:
    Serial.begin(115200);
    //.....................................................................................................//
    pinMode(fire_avail_pin,OUTPUT);
    pinMode(esp_now_avail_pin,OUTPUT);
    pinMode(initi_avail_pin,OUTPUT);

    // Set ESP32 as a Wi-Fi Station
    WiFi.mode(WIFI_STA);

    // Initilize ESP-NOW
    if (esp_now_init() != ESP_OK) {
      Serial.println("Error initializing ESP-NOW");
      digitalWrite(initi_avail_pin, HIGH);
      return;
    }else
    digitalWrite(initi_avail_pin, LOW);

    // Register the send callback
    esp_now_register_send_cb(OnDataSent);

    // Register peer
    memcpy(peerInfo.peer_addr, broadcastAddress, 6);/*to copy specified bytes of memory from source address(broadcastAddres)
                                                      to destination address(peerInfo.peer_addr).*/
    peerInfo.channel = 0;
    peerInfo.encrypt = false;

    // Add peer
    if (esp_now_add_peer(&peerInfo) != ESP_OK){
      Serial.println("Failed to add peer");
      digitalWrite(initi_avail_pin, HIGH);
      return;
    }else
    digitalWrite(initi_avail_pin, LOW);
    //.....................................................................................................//



    //comment out the below line to start inference immediately after upload
    while (!Serial);
    Serial.println("Edge Impulse Inferencing");
    if (ei_camera_init() == false) {
        ei_printf("Failed to initialize Camera!\r\n");
        digitalWrite(initi_avail_pin, HIGH);
    }
    else {
        ei_printf("Camera initialized\r\n");
        digitalWrite(initi_avail_pin, LOW);
    }

    ei_printf("\nStarting continious inference in 2 seconds...\n");
    ei_sleep(2000);
}

/**
* @brief      Get data and run inferencing
*
* @param[in]  debug  Get debug info if true
*/
void loop()
{
    if(frame<0)frame = 0;
    // instead of wait_ms, we'll wait on the signal, this allows threads to cancel us...
    if (ei_sleep(5) != EI_IMPULSE_OK) {
        return;
    }

    snapshot_buf = (uint8_t*)malloc(EI_CAMERA_RAW_FRAME_BUFFER_COLS * EI_CAMERA_RAW_FRAME_BUFFER_ROWS * EI_CAMERA_FRAME_BYTE_SIZE);

    // check if allocation was successful
    if(snapshot_buf == nullptr) {
        ei_printf("ERR: Failed to allocate snapshot buffer!\n");
        digitalWrite(initi_avail_pin, HIGH);
        return;
    }else
    digitalWrite(initi_avail_pin, LOW);

    ei::signal_t signal;
    signal.total_length = EI_CLASSIFIER_INPUT_WIDTH * EI_CLASSIFIER_INPUT_HEIGHT;
    signal.get_data = &ei_camera_get_data;
```

```
    if
(ei_camera_capture((size_t)EI_CLASSIFIER_INPUT_WIDTH,
(size_t)EI_CLASSIFIER_INPUT_HEIGHT, snapshot_buf) ==
false) {
        ei_printf("Failed to capture image\r\n");

        digitalWrite(initi_avail_pin, HIGH);
        free(snapshot_buf);
        return;
    }else
    digitalWrite(initi_avail_pin, LOW);

    // Run the classifier
    ei_impulse_result_t result = { 0 };

    EI_IMPULSE_ERROR err = run_classifier(&signal,
&result, debug_nn);
    if (err != EI_IMPULSE_OK) {
        ei_printf("ERR: Failed to run classifier
(%d)\n", err);

        digitalWrite(initi_avail_pin, HIGH);
        return;
    }else
    digitalWrite(initi_avail_pin, LOW);

    // print the predictions
    // ei_printf("Predictions (DSP: %d ms.,
Classification: %d ms., Anomaly: %d ms.): \n",
    //            result.timing.dsp,
result.timing.classification, result.timing.anomaly);

#if EI_CLASSIFIER_OBJECT_DETECTION == 1
    bool bb_found = result.bounding_boxes[0].value >
0;
    float pred_val = 0;
    for (size_t ix = 0; ix <
result.bounding_boxes_count; ix++) {
        auto bb = result.bounding_boxes[ix];
        if (bb.value == 0) {
            //digitalWrite(fire_avail_pin, LOW);
            continue;
        }
        if(pred_val<bb.value)
        {
          digitalWrite(fire_avail_pin, HIGH);
            myData.cam_info[0] = cam_id;
            myData.cam_info[1] = frame++;

            myData.fire_avail = true;
            myData.predic_score = bb.value;
            myData.x_ = bb.x;
            myData.y_ = bb.y;
            myData.width_ = bb.width;
            myData.height_ = bb.height;

            pred_val = bb.value;
        }
        //ei_printf("    %s (%f) [ x: %u, y: %u,
width: %u, height: %u ]\n", bb.label, bb.value, bb.x,
bb.y, bb.width, bb.height);
    }
    if (!bb_found) {
        //ei_printf("    No objects found\n");

        digitalWrite(fire_avail_pin, LOW);

        myData.cam_info[0] = cam_id;
        myData.cam_info[1] = frame++;
        myData.fire_avail = false;

    }
      //.....................................
.....................................................
.//
    esp_err_t result_esp_now =
esp_now_send(broadcastAddress, (uint8_t *) &myData,
sizeof(myData));

    if (result_esp_now == ESP_OK) {
      Serial.println("Sending confirmed");
      digitalWrite(esp_now_avail_pin, LOW);
    }
    else {
      Serial.println("Sending error");
      digitalWrite(esp_now_avail_pin, HIGH);
    }

     //.....................................
.....................................................
//
#else
    for (size_t ix = 0; ix <
EI_CLASSIFIER_LABEL_COUNT; ix++) {
        ei_printf("    %s: %.5f\n",
result.classification[ix].label,
                                    result.classificati
on[ix].value);
    }
#endif

#if EI_CLASSIFIER_HAS_ANOMALY == 1
        ei_printf("    anomaly score: %.3f\n",
result.anomaly);
#endif



    free(snapshot_buf);

}

/**
 * @brief   Setup image sensor & start streaming
 *
 * @retval  false if initialisation failed
 */
bool ei_camera_init(void) {

    if (is_initialised) return true;

#if defined(CAMERA_MODEL_ESP_EYE)
  pinMode(13, INPUT_PULLUP);
  pinMode(14, INPUT_PULLUP);
#endif

    //initialize the camera
```

```
    esp_err_t err = esp_camera_init(&camera_config);
    if (err != ESP_OK) {
      Serial.printf("Camera init failed with error
0x%x\n", err);
      return false;
    }

    sensor_t * s = esp_camera_sensor_get();


  //Buttons for disableing AEC sensor and AEC DSP
  s->set_exposure_ctrl(s,0); //AEC sensor
  s->set_aec2(s,0); //AEC DSP

  //set value for AE Level and Exposer:
  s->set_ae_level(s,-1);
  s->set_aec_value(s,200);

  s->set_brightness(s,-2);


    if (s->id.PID == OV3660_PID) {
      s->set_vflip(s, 1); // flip it back
      s->set_brightness(s, 1); // up the brightness
just a bit
      s->set_saturation(s, 0); // lower the
saturation
    }

#if defined(CAMERA_MODEL_M5STACK_WIDE)
    s->set_vflip(s, 1);
    s->set_hmirror(s, 1);
#elif defined(CAMERA_MODEL_ESP_EYE)
    s->set_vflip(s, 1);
    s->set_hmirror(s, 1);
    s->set_awb_gain(s, 1);
#endif

    is_initialised = true;
    return true;
}

/**
 * @brief      Stop streaming of sensor data
 */
void ei_camera_deinit(void) {

    //deinitialize the camera
    esp_err_t err = esp_camera_deinit();

    if (err != ESP_OK)
    {
        ei_printf("Camera deinit failed\n");
        return;
    }

    is_initialised = false;
    return;
}


/**
 * @brief      Capture, rescale and crop image
 *
 * @param[in]  img_width     width of output image
 * @param[in]  img_height    height of output image
 * @param[in]  out_buf       pointer to store output
image, NULL may be used
 *                           if ei_camera_frame_buffer
is to be used for capture and resize/cropping.
 *
 * @retval     false if not initialised, image
captured, rescaled or cropped failed
 *
 */
bool ei_camera_capture(uint32_t img_width, uint32_t
img_height, uint8_t *out_buf) {
    bool do_resize = false;

    if (!is_initialised) {
        ei_printf("ERR: Camera is not
initialized\r\n");
        return false;
    }

    camera_fb_t *fb = esp_camera_fb_get();

    if (!fb) {
        ei_printf("Camera capture failed\n");
        return false;
    }

   bool converted = fmt2rgb888(fb->buf, fb->len,
PIXFORMAT_JPEG, snapshot_buf);

   esp_camera_fb_return(fb);

   if(!converted){
       ei_printf("Conversion failed\n");
       return false;
   }

    if ((img_width != EI_CAMERA_RAW_FRAME_BUFFER_COLS)
        || (img_height !=
EI_CAMERA_RAW_FRAME_BUFFER_ROWS)) {
        do_resize = true;
    }

    if (do_resize) {
        ei::image::processing::crop_and_interpolate_rg
b888(
        out_buf,
        EI_CAMERA_RAW_FRAME_BUFFER_COLS,
        EI_CAMERA_RAW_FRAME_BUFFER_ROWS,
        out_buf,
        img_width,
        img_height);
    }


    return true;
}

static int ei_camera_get_data(size_t offset, size_t
length, float *out_ptr)
{
    // we already have a RGB888 buffer, so
recalculate offset into pixel index
    size_t pixel_ix = offset * 3;
```

```
    size_t pixels_left = length;
    size_t out_ptr_ix = 0;

    while (pixels_left != 0) {
        out_ptr[out_ptr_ix] = (snapshot_buf[pixel_ix]
<< 16) + (snapshot_buf[pixel_ix + 1] << 8) +
snapshot_buf[pixel_ix + 2];

        // go to the next pixel
        out_ptr_ix++;
        pixel_ix+=3;
        pixels_left--;
    }
    // and done!
    return 0;
}

#if !defined(EI_CLASSIFIER_SENSOR) ||
EI_CLASSIFIER_SENSOR != EI_CLASSIFIER_SENSOR_CAMERA
#error "Invalid model for current sensor"
#endif
```

Table: *Source Code for the main program*

4 Implementation

4.1 Description

Stereoscopic Imaging unit:

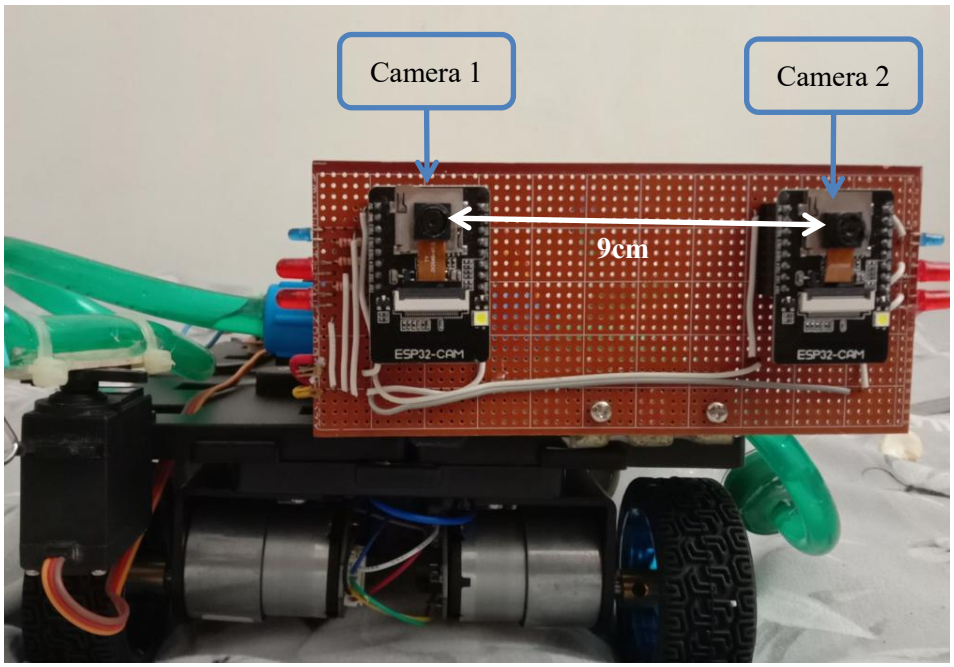


Figure: Stereoscopic Imaging unit(Hardware setup)

Here,we used veroboard to mount both cameras within a per-defined distance.Then trigonometric formulation is used to evaluate the distance from fire source.This procedure is not the most accurate but gives us a somewhat approximate estimation of the distance.

Main Controll unit:

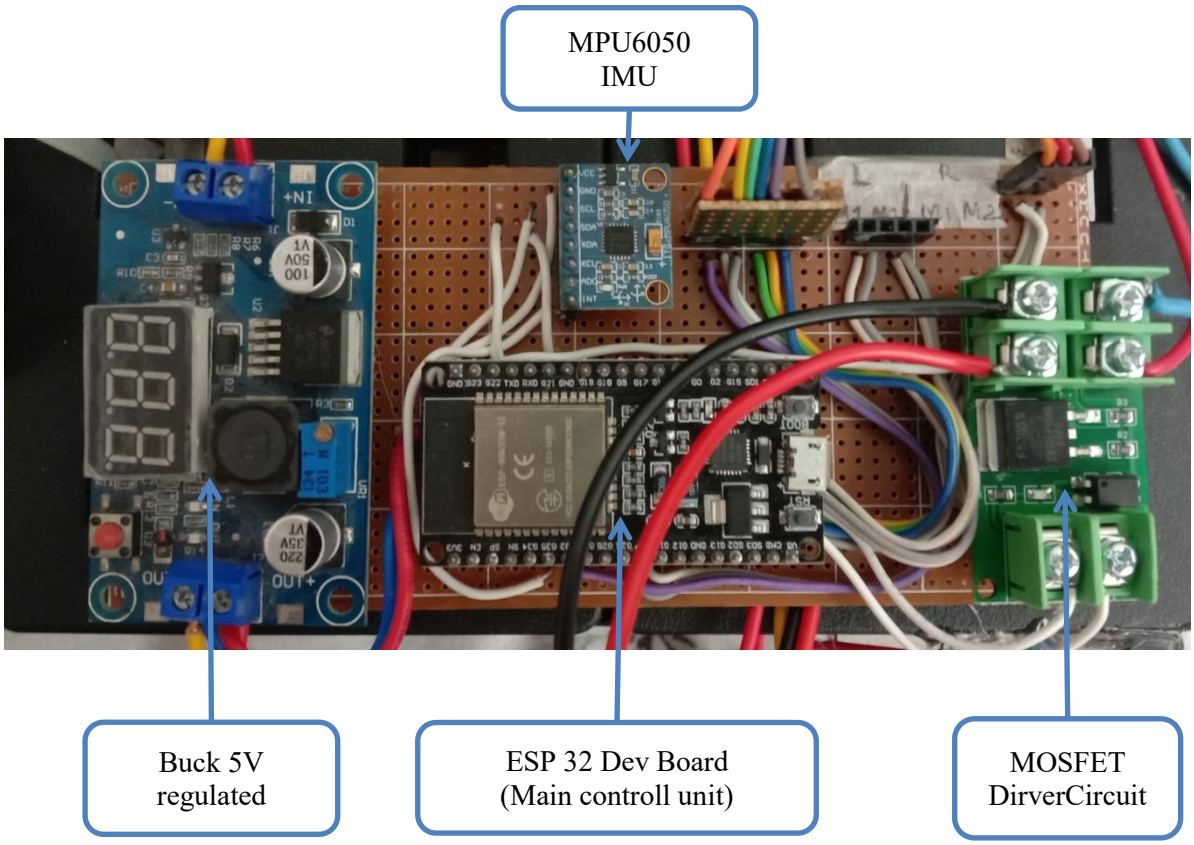


Figure: Main Controller unit(Hardware setup)

Power Management System:

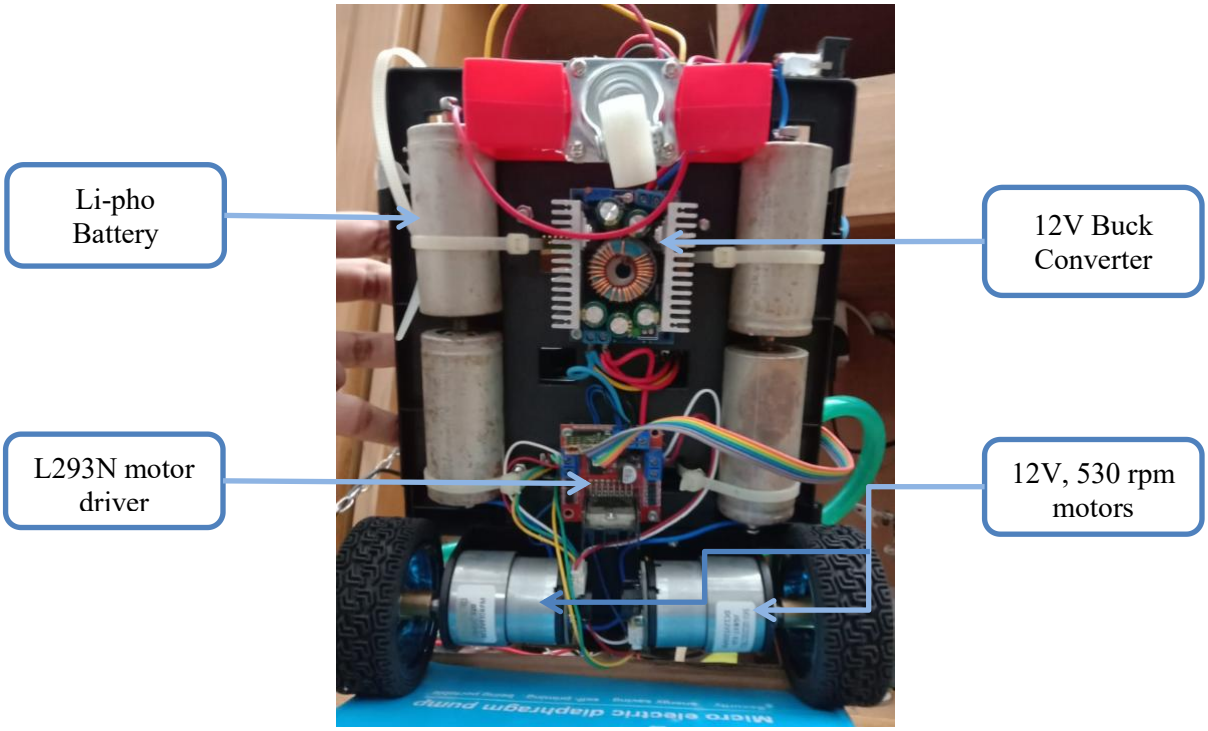


Figure: Power management unit(Hardware setup)

Pump and overall configuration:

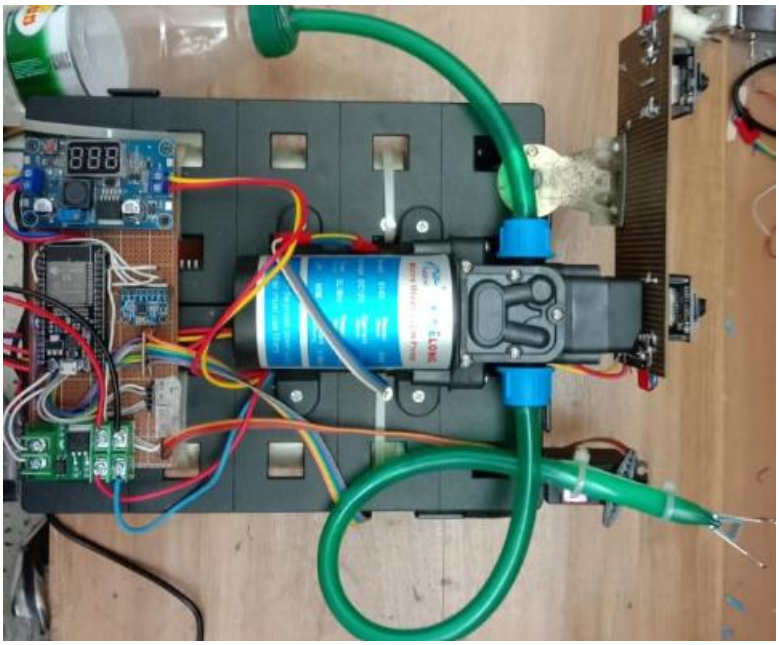


Figure: Pump and overall configuration(Hardware setup)

4.2 Experiment and Data Collection

For the image processing part we collected over 220+ sample images. Then during the classification process split it into 20%-80% partition. The neural network architecture has 27,648 features.
This model runs on the ESP32 cam board with the performance stated below:

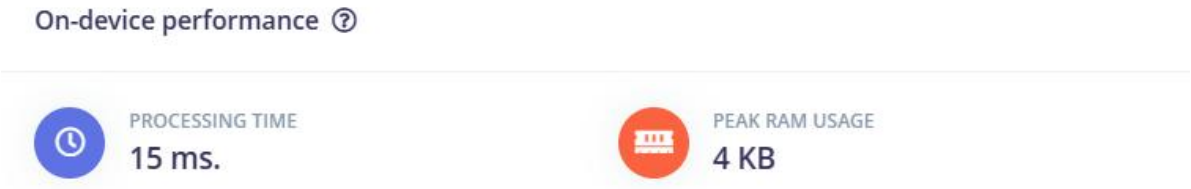


Figure :On device performance (edge impulse)

Now we explore the the data for PID tuner:

We used a combination of Ziegler-Nichos tuning and manual tuning method to achieve a desired PID parameters.first we increase proportional gain K_p until the system reaches an oscillatory state.We then acquire the data through USB port at baud rate of 115200.The graph is plotted and value of K_u and t_u is determined from the graph.

The graph obtained through serial analyzer is given below:

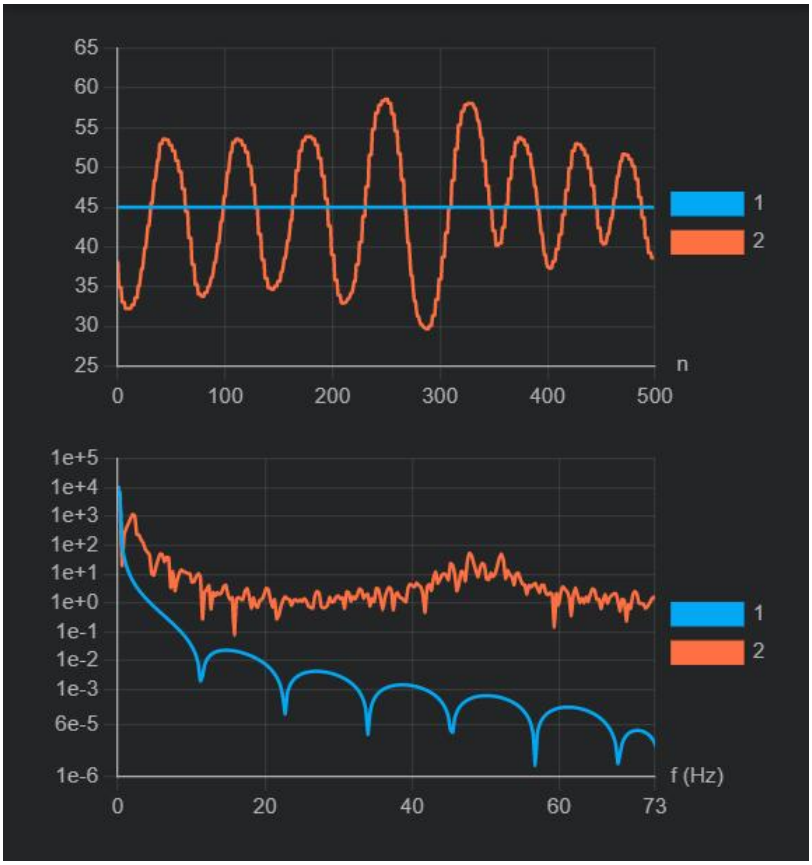


Fig : Response curve of the system during oscillatory state.

Now we will use this data to obtained a tuned PID controller

4.3 Data Analysis

The second graph above is DFT of the output response, this helps us to determine the value of t_u and apply it in Ziegler-Nichos method. After incorporating manual adjustment of PID parameters , we obtained the value of k_p ,k_i ,k_d given below

	Ziegler-Nichos	**Manual Adjustment**
K_p	5	2.9
K_d	0.0075	0.03
K_i	1/300	1/120000

After the manual adjustment our rover body is much more stable. But there are room for improvements in regards to this controller.

4.4 Results

Our rover successfully detects fire.(Although there are cases of false detection mainly due to the reflectivity of surrounding objects).The capability of holding angular position of our rover is sufficiently accurate. But problem arises when it is to move in the forward direction ,some minor oscillatory behaviour is observed. Finally sometimes the nozzles misfires hence extinguishing the fire with maximum efficiency is not achieved.

5 Design Analysis and Evaluation

5.1 Novelty

We were able to run an ML model using just a 248kB micro-controller.Hence we were able to develop and deploy an optimized Machine learning algorithm for our project.The codes for PID controller was built from scratch,hence we were able to deploy a discrete time PID controller.
In our project we transferred the image processing data by defining a custom data structure that incorporates both the camera ID and image frame no. We were able to obtain the DMP(Digital Motion Processing) Data directly from MPU650 .

5.2 Design Considerations

5.2.1 Considerations to public health and safety

Our project will have a profound impact on public health and safety by providing an innovative and reliable solution for early fire detection and rapid response. By swiftly identifying and responding to fire incidents, it will reduce the risk of injury, loss of life, and property damage. Furthermore, the system's autonomous capabilities and accuracy will enhance overall disaster management, ensuring timely interventions to mitigate the devastating consequences of fires in both indoor and outdoor environments, thereby safeguarding communities and public well-being.

5.2.2 Considerations to environment

Our project aligns with important environmental considerations by aiding in the early detection and containment of fires. By swiftly identifying and responding to fire incidents, the Image Processing-Based Fire-Bot can reduce the ecological impact of wildfires, limiting their spread and severity. Additionally, by promoting more efficient firefighting practices, it minimizes the environmental consequences of large-scale fire suppression efforts, such as water and resource consumption. Overall, the project contributes to the preservation of ecosystems and resources, fostering a more sustainable approach to wildfire management.

5.2.3 Considerations to cultural and societal needs

Our project contributes to cultural and societal needs by bolstering fire safety measures. It protects lives, property, and natural resources, fostering a safer environment for communities. Additionally, it aligns with broader sustainability goals by reducing the ecological impact of wildfires.

5.3 Limitations of Tools

- We used a low processing power board because more advanced boards such as raspberry pi were not in stock in the market.
- Our ML model is currently trained only on candle lit fires or similar types of single source fires.
- Currently our rover only works and detects fire it is within 1 meter of the imaging sensor.
- The frame rate of our detection process is extremely low, hence overall our rover works comparatively slowly.
- Our stereoscopic vision algorithm is somewhat rudimentary,hence sometimes it may detect the distance with low accuracy ,which may lead to some unwanted accidents.

5.4 Impact Assessment

5.4.1 Assessment of Societal and Cultural Issues

Autonomous fire fighting bot elimintes the risk of an fire-fighter hence reducing causality rate in such assessments.

5.4.2 Assessment of Health and Safety Issues

Our project will rigorously assess health and safety issues by implementing robust safety protocols and fail-safe mechanisms. It will prioritize the protection of human lives and the Fire-Bot's safe operation in hazardous fire scenarios, minimizing risks and ensuring operational integrity.

5.4.3 Assessment of Legal Issues

Our project will ensure compliance with regulations related to autonomous systems, privacy concerns regarding data collection, and liability considerations in the event of property damage or injury during fire response operations.

5.5 Sustainability and Environmental Impact Evaluation

The sustainability and environmental impact evaluation for our project are critical. We aim to minimize environmental harm by using energy-efficient components, reducing waste in production, and employing eco-friendly materials. The Fire-Bot's swift response can limit fire damage, reducing long-term environmental restoration efforts. Additionally, the system's ability to detect fires in their early stages may lead to reduced emissions from uncontrolled fires. Sustainability is at the forefront, and we will conduct a comprehensive life-cycle assessment to quantify our project's environmental benefits, ensuring that it aligns with ecological preservation and conservation efforts.

5.6Ethical Issues

The development of our Image Processing-Based Fire-Bot project is deeply rooted in ethical principles that guide its design, implementation, and operation. Several ethical challenges were encountered during its development, and proactive measures were taken to mitigate these issues.

Privacy Concerns: The project involves capturing images and data, potentially infringing on individual privacy. Mitigation - To address this concern, we ensure data collected is anonymized and only used for fire detection purposes. Privacy protocols adhere to legal requirements and respect individuals' rights.

Safety and Human Well-being: The autonomous nature of the Fire-Bot raises concerns about its potential impact on human safety. Mitigation - We prioritize safety through robust engineering, rigorous testing, and safety protocols to minimize the risk of harm to humans or property during Fire-Bot operations.

Data Security: Safeguarding sensitive data from potential breaches. Mitigation - We employ robust encryption and cybersecurity measures to protect data integrity and prevent unauthorized access.

By proactively addressing these ethical challenges, our project not only aligns with ethical principles but also enhances its credibility, trustworthiness, and overall positive impact on society, promoting a safer and more sustainable future.

6 Reflection on Individual and Team work

6.1 Individual Contribution of Each Member

1906109 : Stereoscopic camera module synchronization, and modification .
1906110 : Hardware acquiring , tuning PID parameters and comparing response Data.
1906111 : Further improving the rover body by calculating its stress points and modifying its structure.
1906112 : Use the stereoscopic camera module developed by 1906109 to Build and train a FOMO machine learning model.

1906118 : Acquiring proper hardware and finding better alternatives to the current system.
1206125: Manage the power supply system of the rover.

6.2 Mode of TeamWork

We tried work synchronously to improve the efficiency of our team.Our main target was to utilize all the resources that we within this short period of time.At first we divide the project in 2 distinct segments , Machine learning and odometric control. Then each of our team members individually researched their designated topics. Few of our members were assigned to acquire market status of each individual components. And we quickly noticed that there was a shortage in SBC(Single Board Computers). So we were forced to use a micro controller to run the machine learning algorithms.This approach has somewhat reduced the accuracy,but in the end it gave us the desired result .

6.3 Diversity Statement of Team

During the assignment of individual tasks we searched for particular expertise and previous experiences.All of our team mates communicated freely with each other and expressed their concerned regarding any approaching issue.

6.4 Log Book of Project Implementation

Date	Milestone achieved	Individual Role	Team Role	Comments
July 26	Image Processing platform was finalized	1906112		
July 28	Stereoscopic camera setup was build	1906109		
August 2	Data synchronization between Between the imaging sensor & main control board was achieved		Whole team	Successfully achieved desired result
August 3	Power management system was developed	1206125		
August 5	The main rover body components finalized & building process initiated	1906111		
August 7	Sample image collection & training of the FOMO model was done	1906110, 1906112		
August 15	Rover body PID controller code written & tuned	1906110		Was a somewhat iterative process
August 21	Various modules software code unified and synchronized		Whole team	
September 1	The water spray and pump system developed	1906118		
September 4	Final modification and imrovement in odometry of the rover body		Whole team	Met expectation

7 Communication

7.1 Executive Summary

Our project, the Image Processing-Based Fire-Bot, pioneers an advanced solution for early fire detection and response. Leveraging cutting-edge technologies like computer vision, machine learning, thermal imaging, and robotics, the Fire-Bot autonomously identifies fires in real-time, aiding in rapid intervention. Ethical considerations, including privacy, safety, and environmental impact, are at the core of our design. We prioritize transparency, accountability, and community involvement to ensure the project's ethical alignment. The Fire-Bot's potential to mitigate the devastating effects of wildfires and indoor fires underscores its significance in enhancing fire safety and disaster management.

7.2 User Manual

The robot will autonomously detect and extinguish fire. Before this the robot must be initialized accordingly.

- Firstly the the water container must be filled.
- Then the container must be attached and the air release valve must be opened.
- Then Power port must be connected.
- The rover body must be placed in a flat surface for at least 2 seconds.
- Then the rover will automatically starts its detection and execution routine.
- In order to turn it off ,just unplug its power port.

8 Project Management and Cost Analysis

8.1 Bill of Materials

Item	**Quantity**	**Cost per item(BDT)**	**Total cost(BDT)**
Esp32 dev board	1	690	690
Esp32 camera	2	800	1600
Motor (12V, 530rpm)	2	1100	2200
Motor driver(L293N)	1	195	195
Buck converter(12A rated)	1	500	500
Buck converter(3A rated)	1	120	120
Pump (DIAPHRAGM WATER PUMP)	1	1500	1500
Chassis material	1	500	500
Battery(Li-pho used)	4	200	800
Miscellaneous	-	1000	1000
Total			**9150**

9 Future Work

- Futute works includes improving the Machine learning model to get higher accuracy and also to detect different types of fire
- Improve to better hardware for faster response.
- Increase water holding capacity.
- Improve the odometry and complex routing and navigating through various surfaces.
- Increase the frame rate of detection.
- Try to incorporate swarm feature, meaning different fire-fighting modules will communicate with each other to improve their performance.

10 References

[1] "Colleted Database to libraray file generation -edge impulse "https://studio.edgeimpulse.com/studio/262698(july ,2023)
[2] " Ziegler-Nichols method and tuning procedure" - "https://www.mstarlabs.com/control/znrule.html"
[3] "MotionApp source code"- "https://invensense.tdk.com/products/motion-tracking/6-axis/mpu-6050/"
[4] "Data classification -edge impulse" - "https://studio.edgeimpulse.com/studio/262698/classification"
[5] "Esp_Now syntax-ESP IDF" "https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/network/esp_now.html"