**Code Used in Arduino our Project**

#include <Servo.h>

#include <Wire.h>

#include <LiquidCrystal\_I2C.h>

LiquidCrystal\_I2C lcd(0x3F, 20, 2);

Servo servo ;

int eastLDR = A0;

int westLDR = A1;

int angle = 90;

void setup()

{

Serial.begin(9600);

lcd.begin();

servo.attach(9);

pinMode(9,OUTPUT);

pinMode(A0,INPUT); //To collect from LDR1

pinMode(A1,INPUT); //To collect from LDR2

pinMode(A4,OUTPUT); // For LCD display analog pin.

pinMode(A5,OUTPUT); // For LCD display digital pin.

}

void loop()

{

int east = analogRead(eastLDR);

int west = analogRead(westLDR);

Serial.print("eastLDR:");

Serial.println(east);

Serial.print("westLDR:");

Serial.println(west);

lcd.setCursor(0,0);

int error = east - west;

if (error > 15) // No action if the difference is not more than 15

{

if (angle < 180)

{

angle=angle+10;

servo.write(angle);

digitalWrite(13,HIGH);

digitalWrite(12,LOW);

}

//Serial.println("Turning to east");

lcd.print("Turning to east");

}

else if (error < -15) // No action if the difference is not less than -15

{

if (angle > 0)

{

angle=angle-10;

servo.write(angle);

digitalWrite(12,HIGH);

digitalWrite(13,LOW);

}

//Serial.println("Turning to west");

lcd.print("Turning to west");

}

lcd.setCursor(0,1);

lcd.print("Angle:");

lcd.print(angle);

Serial.print("Angle:");

Serial.println(angle);

//lcd.clear();

delay(1000);

}

**Code used for machine learning and impulse**

\* 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.

\*

\*/

/\* Includes ---------------------------------------------------------------- \*/

#include <solartrackerv5\_inferencing.h>

#include <Servo.h>

// Servo motor setup

Servo servoMotor;

const int servoPin = 9;

// LDR pins setup

const int eastLDRPin = A0;

const int westLDRPin = A1;

// Variables to store LDR values

int eastLDRValue = 0;

int westLDRValue = 0;

// Buffer for the features

static float features[EI\_CLASSIFIER\_DSP\_INPUT\_FRAME\_SIZE];

/\*\*

\* @brief Copy raw feature data in out\_ptr

\* Function called by inference library

\*

\* @param[in] offset The offset

\* @param[in] length The length

\* @param out\_ptr The out pointer

\*

\* @return 0

\*/

int raw\_feature\_get\_data(size\_t offset, size\_t length, float \*out\_ptr) {

memcpy(out\_ptr, &features[offset], length \* sizeof(float));

return 0;

}

void print\_inference\_result(ei\_impulse\_result\_t result);

// Function to get analog values from LDRs

void get\_analog\_values() {

// Read LDR values

eastLDRValue = analogRead(eastLDRPin);

westLDRValue = analogRead(westLDRPin);

// Update features array with LDR values

features[0] = static\_cast<float>(eastLDRValue);

features[1] = static\_cast<float>(westLDRValue);

// Print LDR values on Serial Monitor

Serial.print("East LDR Value: ");

Serial.print(eastLDRValue);

Serial.print("\tWest LDR Value: ");

Serial.println(westLDRValue);

}

/\*\*

\* @brief Arduino setup function

\*/

void setup()

{

// put your setup code here, to run once:

Serial.begin(115200);

// comment out the below line to cancel the wait for USB connection (needed for native USB)

while (!Serial);

Serial.println("Edge Impulse Inferencing Demo");

// Servo motor setup

servoMotor.attach(servoPin);

// Set initial servo motor angle to 0 degrees

servoMotor.write(0);

// LDR setup

pinMode(eastLDRPin, INPUT);

pinMode(westLDRPin, INPUT);

}

/\*\*

\* @brief Arduino main function

\*/

void loop()

{

ei\_printf("Edge Impulse standalone inferencing (Arduino)\n");

// Get analog values from LDRs

get\_analog\_values();

ei\_impulse\_result\_t result = { 0 };

// the features are stored into flash, and we don't want to load everything into RAM

signal\_t features\_signal;

features\_signal.total\_length = EI\_CLASSIFIER\_DSP\_INPUT\_FRAME\_SIZE;

features\_signal.get\_data = &raw\_feature\_get\_data;

// invoke the impulse

EI\_IMPULSE\_ERROR res = run\_classifier(&features\_signal, &result, false /\* debug \*/);

if (res != EI\_IMPULSE\_OK) {

ei\_printf("ERR: Failed to run classifier (%d)\n", res);

return;

}

// print inference return code

ei\_printf("run\_classifier returned: %d\r\n", res);

print\_inference\_result(result);

// Find the predicted class with the maximum value

int maxClassIndex = 0;

float maxValue = result.classification[0].value;

for (int i = 1; i < EI\_CLASSIFIER\_LABEL\_COUNT; i++) {

if (result.classification[i].value > maxValue) {

maxValue = result.classification[i].value;

maxClassIndex = i;

}

}

// Move the servo motor based on the predicted class with the maximum value

int angle = map(maxClassIndex, 0, EI\_CLASSIFIER\_LABEL\_COUNT - 1, 0, 180);

servoMotor.write(angle);

// Print the servo motor angle after being moved

Serial.print("Servo Motor Angle: ");

Serial.println(angle);

delay(1000);

}

void print\_inference\_result(ei\_impulse\_result\_t result) {

// Print how long it took to perform inference

ei\_printf("Timing: DSP %d ms, inference %d ms, anomaly %d ms\r\n",

result.timing.dsp,

result.timing.classification,

result.timing.anomaly);

// Print the prediction results (object detection)

#if EI\_CLASSIFIER\_OBJECT\_DETECTION == 1

ei\_printf("Object detection bounding boxes:\r\n");

for (uint32\_t i = 0; i < result.bounding\_boxes\_count; i++) {

ei\_impulse\_result\_bounding\_box\_t bb = result.bounding\_boxes[i];

if (bb.value == 0) {

continue;

}

ei\_printf(" %s (%f) [ x: %u, y: %u, width: %u, height: %u ]\r\n",

bb.label,

bb.value,

bb.x,

bb.y,

bb.width,

bb.height);

}

// Print the prediction results (classification)

#else

ei\_printf("Predictions:\r\n");

for (uint16\_t i = 0; i < EI\_CLASSIFIER\_LABEL\_COUNT; i++) {

ei\_printf(" %s: ", ei\_classifier\_inferencing\_categories[i]);

ei\_printf("%.5f\r\n", result.classification[i].value);

}

#endif

// Print anomaly result (if it exists)

#if EI\_CLASSIFIER\_HAS\_ANOMALY == 1

ei\_printf("Anomaly prediction: %.3f\r\n", result.anomaly);

#endif

}