The Fan of the Future

Samskrith Raghav

8th Grade

Abstract

As global warming worsens, people need a way to stay cool while conserving energy. People, especially in poor countries, employ fans to stay cool in an increasingly warmer climate. However, most fans are either fixed or have a predetermined angle of oscillation. This often results in wasted energy, because the fan either does not cover everyone in the room or wastes energy by blowing air to the wall. The goal of my project was to create a relatively inexpensive oscillating table fan which uses artificial intelligence to detect people's faces and adjust the angle of oscillation duly, saving energy and time. The first prototype of this invention uses an Arduino Mega 2560 Circuit Board, an ESP32-CAM with embedded artificial intelligence, a servo motor for oscillation, a DC motor to simulate the actual fan, and four 1.5 V batteries to power the servo and the DC motor. Some additional features which would help optimize the fan are: a sensor to activate the servo and DC motor only when someone is in the room, a temperature sensor which controls the DC motor/fan rotation speed, and a more powerful AI camera to help optimize the code. The fan worked mostly as intended; it could detect faces and oscillate the fan between them. However, it couldn't detect more than two faces at a time, and some hardware limitations prolonged the code and made the processing time much slower. Further iterations of the fan must be made in order to truly achieve the original objective.

Table of Contents

Real World Problem

Table fans or tower fans in the market are either set to a single direction or have a predetermined angle of oscillation. This causes a few problems: either the fan will waste energy by blowing air towards nothing or it fails to cover everyone in the room. This means that the fan is not achieving its intended goal; it is not covering everyone like it is supposed to. The energy wastage of an oscillating fan is also a problem because the extra electricity generated to power the fan equates to more pollution, contributing to the ever worsening problem of global warming.

Engineering Goal

My goal is to create a relatively inexpensive fan which efficiently detects the presence of people and directs the air towards them effectively. This fan calibrates the angle of its oscillation by detecting people's faces using artificial intelligence. ESP32-CAM on the fan periodically takes a picture and sends it to the ESP32-CAM microcontroller, which uses Artificial Intelligence software to detect the faces of people. The program then estimates the location of the person in the photo grid and then calculates the angle of the face from the fan. It then calculates the optimal angle to oscillate and sends instructions to the servo motor (SG90), which carries the fan, to change its position and oscillate. The servo motor oscillates to cover the detected people, conserving both energy and time.

Initial Research

Artificial Intelligence Face Detection

Artificial Intelligence, or AI, is human-like intelligence and ability to learn demonstrated by a computer. The use of AI ranges wide and far, from self-driving cars to computer-aided disease diagnosis. In this project, AI was used to detect human faces. It was integrated into the ESP32-CAM used in the project. Unlike a regular program, which is given the code and the input and delivers the output, the AI takes many inputs and the outputs and uses them to find the rules, which are used to construct the model. The AI then uses the model and any given inputs and gives an output. The model used in this project is called MTMN, or Multi Tasking Mobile Net. MTMN uses 3 "layers", or neural networks, to detect faces. The first net is the P-Net, or the Proposal-Net, which scans the image for all possible faces, puts rectangular boxes around them, and submits them to the second layer. The second net is the R-Net, or the Refine-Net, which narrows down the number of possible faces/boxes to one face and submits that face/box to the third layer. The third net is the O-Net, or the Output-Net, which makes sure the submitted face is really a face and then outputs the coordinates of the box around the final face on the photo. The camera has an integrated coordinate system which is mapped onto the image. The coordinates are used to calculate the angle to oscillate the fan towards a person’s face. Each layer is a neural network, which is a series of algorithms which are used to find patterns, to develop rules for what a face is.

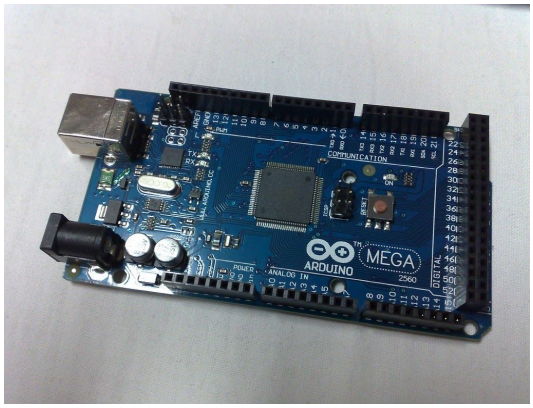
Servo Motor

A Servo Motor is used to oscillate the fan precisely to blow air at the faces detected by the ESP32-CAM. Unlike a regular DC motor, a servo can be programmed to turn left or right, so it is perfect for the precision and adaptability needed for the oscillation of the fan. The frequency of the electrical pulse sent to the servo controlled the angle to which the servo rotates; a different electrical frequency meant a different angle. A standard DC converted the electrical energy into rotation, but a gearbox and a voltage measurement device along with a control circuit work to control the direction and angle of rotation based on the current. However, the servo used in the project can only rotate 180 degrees, making it less than optimal for a marketable product.

The Fan(DC Motor)

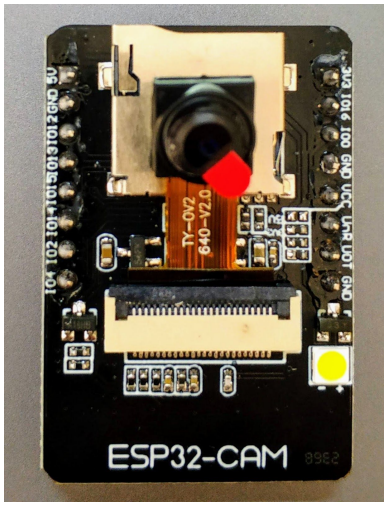
A DC motor was used to simulate a real fan. The 1.5 V battery generates the electricity required for the motor. Within the motor, a wire is placed inside a fixed horseshoe magnet so the wire's magnetic field is within the horseshoe magnet's magnetic field. The interaction of the two magnetic fields causes the wire to start spinning. The direction of the current determines the direction in which the motor rotates. The cork and a set of blades is connected to this rotating motor creating the fan. This fan sits on top of the Servo Motor.

Arduino



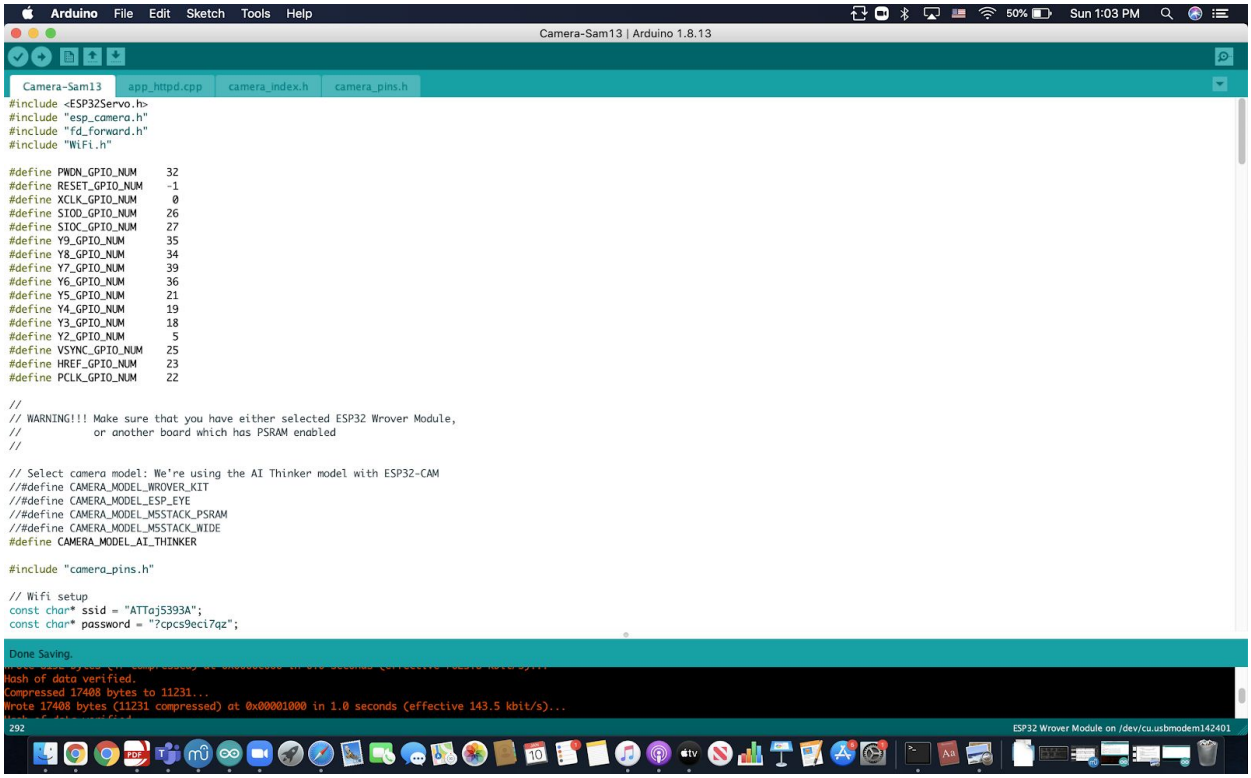
Arduino is an open source circuit board (also called a microcontroller) used by many people to create practical devices, such as robots, keyboards, and LED strips. Arduino can easily be learned and used, and there are plenty of tutorials to be found online. The Arduino takes input from a computer and gives instructions to several devices, such as LEDs, Servos, and DC motors. In this project, the Arduino receives instructions from the computer and passes it to the ESP32.

ESP32-CAM micro controller with a camera



The ESP32-CAM micro controller has two main functions in this project: face detection and servo motor oscillation control. The ESP32-CAM has a processor which runs the software program that detects faces and controls the servo motor written for this project. The software used by the camera is called ESP-WHO, and the facial detection and recognition is part of the esp-face component of ESP-WHO. The Camera itself uses a lens to divert the light rays to a light-sensitive material. The light sensitive material is used to generate a pixelated image in which each pixel is assigned a value and a color known as the RGB value. The image is then given to the integrated MTMN model, which detects any human faces as detailed in the Artificial Intelligence and Face Detection section. In the project, the coordinates of the face are used by the ESP32-CAM to calculate the angle which the servo has to turn and send instructions to the servo to oscillate at that angle.

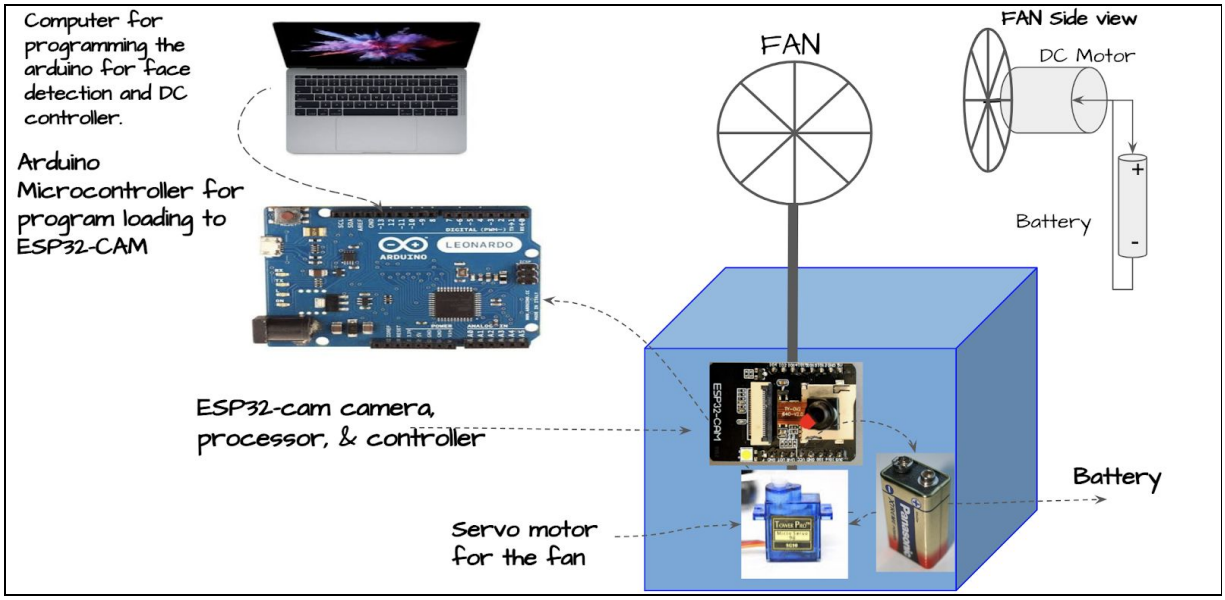
Arduino IDE



Arduino Integrated Development Environment, or Arduino IDE, is an application which programmers can use to code any Arduino compatible boards, such as boards branded Arduino and Elegoo. The programming language is a C/C++ “dialect” which is easy to learn and master.

Device Development

The goal of the project was to design a working prototype of the fan which could detect multiple faces and adapt to oscillate the fan and blow the air at all people present. The basic diagram of the fan, micro controller and the programming setup is given below:



The fan is designed with a DC motor, a cork and handmade blades. The application used to code the ESP32-CAM was Arduino IDE, and the language used to code it was an Arduino language. Arduino IDE was chosen for its ease of learning and simple syntax. The SG90 Servo was used due to its low cost. Arduino Mega 2560 was used due to its relatively low cost and numerous applications. The ESP32-CAM was used due to its relatively low cost, camera quality and for its sophisticated AI package.

Materials

- 1 Aluminium Metal sheet-used to make fan blades and base of the fan
- 12 Breadboard Jumper Wires-used to connect the parts of the fan
- 1 Elegoo Solderless Breadboard used to organize the wires
- Tape-used to connect the DC motor to the servo
- 1 DC(Direct Current) Motor-used to create the fan rotation

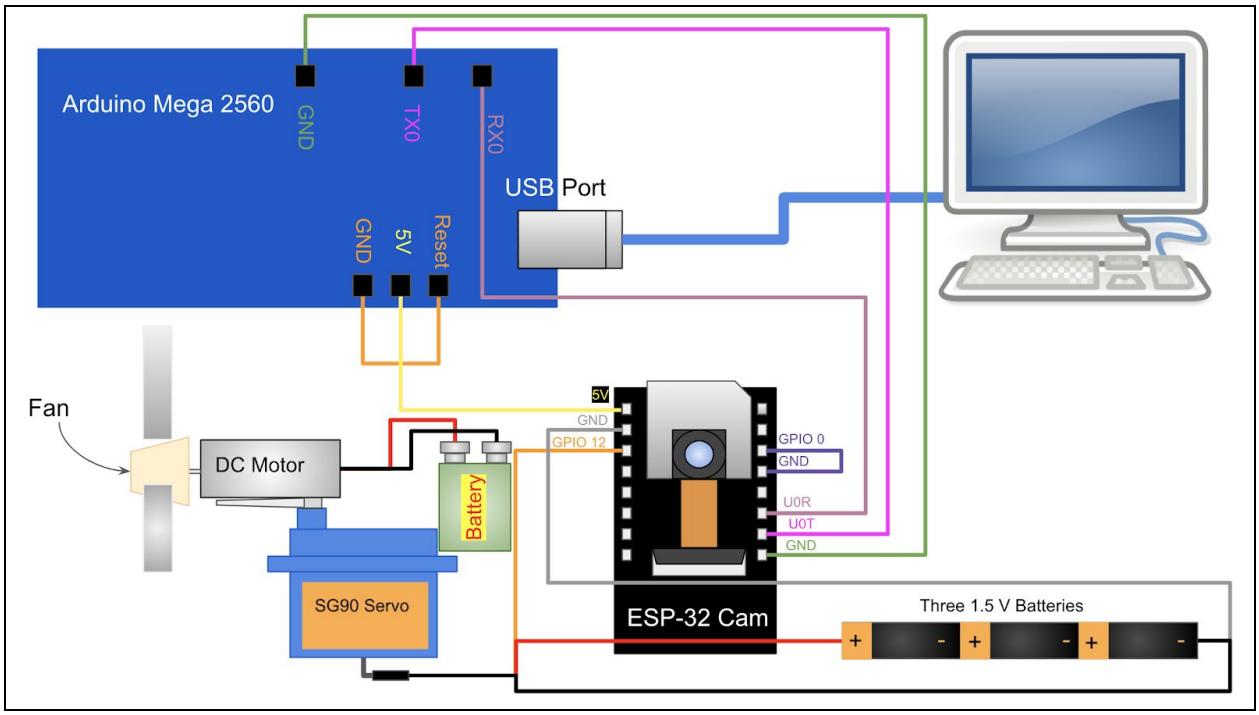
- Arduino or Elegoo Mega2560 MicroController (hardware)-used to connect the ESP32-CAM to Arduino IDE
- Arduino Integrated Development Environment (software)-used to code the fan
- ESP32-CAM Camera microcontroller with 2 megapixel camera-used to detect faces and calculate the angle the servo should turn
- SG90 Servo Motor-used to oscillate the fan
- 1 Cork-used to connect the fan blades to the DC motor
- Four 1.5V batteries-used to power the servo and DC Motor
- 1 computer-used to code the fan and power the ESP32-CAM

Methods

Design and Construction

1. Cut and twist the metal sheet into a cylindrical tower.
2. Cut the remainder of the metal sheet into circles.
3. Use servo screws included with the servo to screw the SG90 servo motor to one of the circles.
4. Cut out a rectangular prism from the metal sheet and make it the base of the fan.
5. Glue the circles to the top and bottom of the cylinder.
6. Use tape to attach the DC motor horizontally at the top of the motor.
7. Take the cork and use the DC motor to puncture it in the middle
8. Cut out fan blades from the metal sheet and attach them to the cork. Make sure to attach the blades at an angle.

9. Use tape to secure the ESP32-CAM to the base of the fan.
10. Use the wires and the breadboard to connect the ESP32-CAM to the arduino/elegoo, the elegoo to the computer to transmit the code, the servo to the ESP32-CAM and the three 1.5 V batteries, and the 9V battery to the DC motor. (Make sure to align the ESP32-CAM with the servo; otherwise, the angle will be incorrect.) Here is the schematic diagram:



11. Use Arduino IDE to write a program to take pictures of the environment and oscillate the fan. Here is the pseudocode for the program:
    a. Detect faces.
        i. Take a picture with the ESP32-CAM.
        ii. Attempt to detect a face in the picture.
        iii. Repeat steps i-ii nine more times to detect all the faces.
        iv. If faces detected:
            1. Get the x-coordinates for all the faces and move on to step b.

v. If no faces detected:

1. Go back to step i.

b. Calculate the angle between the leftmost and rightmost faces.

i. Convert the coordinates of the faces into angles.

ii. Make sure to test if the camera is flipped or not and adjust the algorithm accordingly.

c. Make the servo oscillate between the two angles 10 times.

d. Go back to step a.

Testing

1. Seat one person on a couch with no objects surrounding them.
2. Turn on the fan and let it oscillate.
3. Check the oscillation to see if it pointed at the person.
4. Seat at least two people on the couch with multiple inanimate objects(no paintings of people) around them.
5. Turn on the fan and let it oscillate.
6. Check the oscillation to see if it oscillated between the two outermost people without oscillating further.
7. Seat the two people in different positions and test the results.
8. Repeat step 6 three more times and record the results.

Precautions

- Keep water away from electronics.

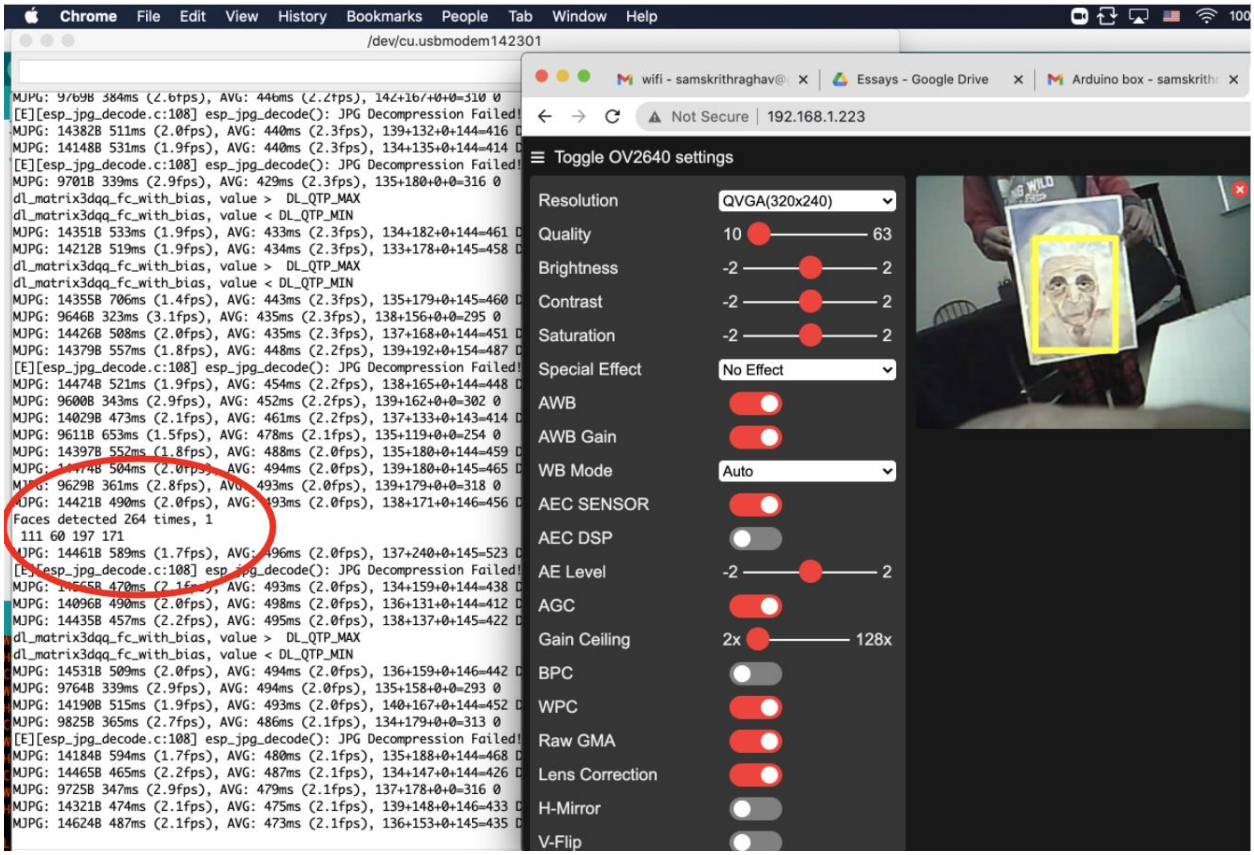
- Work with the metal sheet only with adult supervision.
- Unplug the Arduino immediately if anything goes wrong.

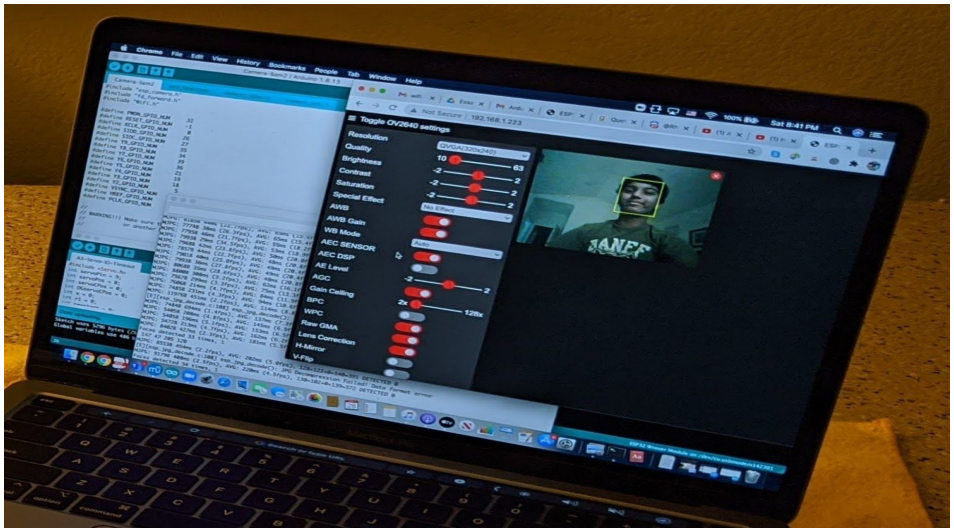
Challenges Overcome

Several problems were encountered during this project. Firstly, the workings of the Arduino, ESP32-CAM, ESP-WHO face detection software, and servo motor had to be learned individually and made to work together. The inventor had to view many videos and articles on each subject until it was thoroughly understood how each component of the fan worked. Next, The ESP32-CAM was unable to detect multiple faces simultaneously; however, detecting multiple faces was important to the project, so an alternative method had to be developed. The prototype built takes ten pictures and scans each input for a face in order to maximize the chance of detecting every face. Additionally, The servo was unable to support the weight of the fan, so a base had to be built to support it. Unfortunately, due to the global COVID-19 pandemic, there was no way to test the fan with more than three people, so print-outs and paintings of people's faces were used to simulate people. The ESP-WHO module only had limited information on using the (x,y) coordinates of the face information it sends. Several test programs were written to determine how the information is returned from the face detection. The ESP32-CAM and the open source software support for it were still relatively new at the time the prototype was being made, so the hardware and software were somewhat difficult to work with, and there were few tutorials online which could help. Finally, because of the time spent writing and fine-tuning the program, there was not enough time to construct the base described in the "Methods" section, so the makeshift base used for testing caused some problems.

Results

The device was able to take multiple images and recognize multiple faces using the workaround developed; however, an image output was not shown in order to speed up the program. Here are some sample images from a previous version of the program which did output an image to the user:





However, the power of the camera and the sophistication of the technology made it almost impossible for the AI to detect more than two faces at once, even from a short distance. However, the program, servo, and DC motor worked well. The program detected the two faces, and the fan did rotate accordingly. However, further testing must be done to see the limits of the first prototype of this fan.

Data for Single Face Detection

	Start Angle	End Angle	Improvement against 180° oscillating fan	Difference against 180° oscillating fan
Trial 1	81	98	10.588 times more effective	163 degrees less
Trial 2	80	99	9.473 times more effective	161 degrees less
Trial 3	83	102	9.473 times more effective	161 degrees less
Trial 4	82	99	10.588 times more effective	163 degrees less

Trial 5	80	96	11.25 times more effective	164 degrees less
Average	81.2	98.8	10.22 times more effective	162.4 degrees less

Data for Dual Face Detection

	Start Angle	End Angle	Improvement against 90° oscillating fan	Difference against 90° oscillating fan
Trial 1	69	103	5.294 × more effective	146 degrees less
Trial 2	66	104	4.737 × more effective	142 degrees less
Trial 3	66	91	7.2 × more effective	155 degrees less
Trial 4	68	85	10.588 × more effective	163 degrees less
Trial 5	68	103	5.143 × more effective	145 degrees less
Average	67.4	97.2	6.592 × more effective	150.2 degrees less

Conclusions

The goal of this project was to develop a relatively inexpensive oscillating fan which detects people's faces and adjusts the angle of oscillation accordingly. The testing done shows

that the first prototype of the project was successful; however, more testing must be done to understand the level of success and limits of this fan.

The fan was able to detect multiple faces at the same time using the workaround developed; however, the workaround was still very inconsistent, and often the AI didn't detect any faces at all. Increasing the number of photos taken helped increase the accuracy of the AI, but the time taken for the AI to go over each photo made the wait agonisingly long.

Still, the fan was able to take pictures, detect faces, and adjust the angle of oscillation accordingly, and it did its job surprisingly well. However, adjustment must be made in order for the project to truly achieve its stated purpose.

Suggestions for Improvement and Further Research

To further improve this invention, an infrared sensor could be used in conjunction with the camera to more accurately detect people rather than just posters. The camera itself could be situated on a servo to allow for a wider range of spatial awareness. A better servo could be used to mount the fan in order to allow it to turn 360 degrees rather than the current 180 degrees. A temperature sensor could be added to regulate the fan speed depending on the temperature. Because of the time spent writing and fine-tuning the program, there was not enough time to construct the base described in the "Methods" section, so a second prototype of the fan would include some version of that base. A more sophisticated and powerful camera and AI could be used to more accurately detect faces and get rid of the workaround which substantially lengthened and complicated the code.

Acknowledgements

I would like to thank my parents for their guidance and support during my project. I would especially like to thank my dad, who taught me the basics of C and C++ within less than a month. I would like to thank all the wonderful people who published multiple videos and articles on Arduino, the ESP32-CAM, servo motors, and oscillating fans. Finally, I would like to thank my teacher for approving my choice of this difficult project for my science fair.

References

Tech StudyCell. (2020, June 20). *How to program ESP32 CAM using Arduino UNO* [Video]. Youtube. https://www.youtube.com/watch?v=q-KIpFIbRMk&t=240s

Viral Science. (2020, March 28). *ESP32 CAM Getting Started | Face Detection* [Video]. Youtube. https://www.youtube.com/watch?v=U7qbehy9aDo&t=374s

espressif. (2020, August 31). *ESP-WHO*. Github. Retrieved November 26, 2020 from https://github.com/espressif/esp-who/blob/master/README.md

Dejan. (2018, March 18). *How Servo Motors Work & How To Control Servos using Arduino*. How To Mechatronics. Retrieved November 14, 2020 from https://howtomechatronics.com/how-it-works/how-servo-motors-work-how-to-control-servos-using-arduino/

Rui Santos. (2019, March 18). *ESP32-CAM Video Streaming and Face Recognition with Arduino IDE*. Random Nerd Tutorials. Retrieved November 25, 2020 from https://randomnerdtutorials.com/esp32-cam-video-streaming-face-recognition-arduino-ide/

Augmented Startups. (2017, November 18). *Convolutional Neural Networks - Fun and Easy Machine Learning* [Video]. Youtube. https://www.youtube.com/watch?v=DEcvz_7a2Fs&t

DragonPh. (2020, April 17). *ESP32-CAM: Remote Control Object Detection Camera*. Hackster. Retrieved November 24, 2020 from https://www.hackster.io/dragonph/esp32-cam-remote-control-object-detection-camera-7f18af

Arduino. (2018, February 5). *Arduino Software (IDE)*. Arduino. Retrieved November 25, 2020 from https://www.arduino.cc/en/Guide/Environment

MagLab. (2019, June 17). *DC Motor*. Magnet Academy. Retrieved January 9, 2021 from https://nationalmaglab.org/education/magnet-academy/watch-play/interactive/dc-motor

Program Listing

```
#include <ESP32Servo.h>
#include "esp_camera.h"
#include "fd_forward.h"
#include "WiFi.h"

#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

//
// WARNING!!! Make sure that you have either selected ESP32 Wrover Module,
//            or another board which has PSRAM enabled
//

// Select camera model: We're using the AI Thinker model with ESP32-CAM
//#define CAMERA_MODEL_WROVER_KIT
//#define CAMERA_MODEL_ESP_EYE
//#define CAMERA_MODEL_M5STACK_PSRAM
//#define CAMERA_MODEL_M5STACK_WIDE
#define CAMERA_MODEL_AI_THINKER

#include "camera_pins.h"

// Wifi setup
const char* ssid = "ATTaj5393A";
const char* password = "?cpcs9eci7qz";
void startCameraServer();//wifi unused

mtmn_config_t mtmn_config = {0}; //global variable dec for cam
int detections = 0; //detection variable define

Servo myservo;  // create servo object to control a servo
```

```
/*
   Type: Utility function.
   Purpose: Setup the camera configuration in the ESP32-CAM module.
   Arguments: None
   Returns: Nothing
*/
void setup_camera() { //setup the camera
  camera_fb_t * frame1; //get the pic

  camera_config_t config; // configure the camera

  //configure the camera
  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_UXGA;
    config.jpeg_quality = 10;
    config.fb_count = 2;
  } else {
    config.frame_size = FRAMESIZE_SVGA;
    config.jpeg_quality = 12;
    config.fb_count = 1;
  }

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

```
#endif

  // 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;
  }

  sensor_t * s = esp_camera_sensor_get();
  //initial sensors are flipped vertically and colors are a bit saturated
  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, -2);//lower the saturation
  }
  //drop down frame size for higher initial frame rate
  s->set_framesize(s, FRAMESIZE_QVGA);

#if defined(CAMERA_MODEL_M5STACK_WIDE)
  s->set_vflip(s, 1);
  s->set_hmirror(s, 1);
#endif
  // MTMN is a combination of MTCNN and MobileNets
  // MTCNN: Multi-Task Convolutional Neural Network
  // MTCNN is used for face detection.
  mtmn_config = mtmn_init_config();
}

/*
   Type: Utility function.
   Purpose: Setup the Serial Monitor.
   Arguments: the Serial Monitor baud rate, and whether or not to debug the code
   Returns: Nothing
*/
void setup_serial(int baud_rate, bool debug_code) {
  Serial.begin(baud_rate);
  Serial.setDebugOutput(debug_code);
  Serial.println();
}

/*
   Type: Utility function.
   Purpose: Setup the servo and attach it to the ESP32.
   Arguments: servo_pin
   Returns: Nothing
*/
void setup_servo(int servo_pin) {//servo setup
```

```
  ESP32PWM::allocateTimer(0);
  ESP32PWM::allocateTimer(1);
  ESP32PWM::allocateTimer(2);
  ESP32PWM::allocateTimer(3);
  myservo.setPeriodHertz(50);    // standard 50 hz servo
  myservo.attach(servo_pin); // attaches the servo on pin 18 to the servo object
}

/*
   Type: Utility function.
   Purpose: Convert the x cordinate given into an angle for the servo
   Arguments: x coordinate
   Returns: Nothing
*/
int calc_angle(int x)//calculate angle from coordinate
{
  float a;

  if (x <= 0) {
    a = 0;
  }
  else if (x >= 360) {
    a = 150;
  }
  else if (x == 180) {
    a = 90;
  }
  else if (0 < x < 180) {
    a = 90 + 30 - x * 0.1667 ;
  }
  else if (180 < x < 360) {
    a = 90 + x * 0.1667 - 30 ;
  }
  Serial.printf(" x = %d, NEW SERVO ANGLE + %d\n", x, (int) a);
  return (int) a;
}

/*
   Type: Utility function.
   Purpose: Make the servo do the sweep for whichever values are given.
   Arguments: lp, spos, epos
   Returns: Output to servo
*/
void sweep_servo(int lp, int spos, int epos) {//actual servo sweep command

  int pos;

  for ( ; lp-- > 0; ) {
```

```
    for (pos = spos; pos <= epos; pos += 1) { // goes from 0 degrees to 180 degrees
      // in steps of 1 degree
      myservo.write(pos);    // tell servo to go to position in variable 'pos'
      delay(15);             // waits 15ms for the servo to reach the position
    }
    for (pos = epos; pos >= spos; pos -= 1) { // goes from 180 degrees to 0 degrees
      myservo.write(pos);    // tell servo to go to position in variable 'pos'
      delay(15);             // waits 15ms for the servo to reach the position
    }
  }
}

/*
   Type: Template function.
   Purpose: Call the functions, run the iteration loop, and send output to servo.
   Arguments: None
   Returns: Nothing
*/
void setup() {
  pinMode(33, OUTPUT);
  camera_fb_t * frame1;
  camera_config_t config;

  setup_serial(115200, true);  // setup the computer serial monitor
  setup_servo(12);  // setup the fan
  setup_camera();   // setup the camera

  while (true) {   // forever
    // put your main code here, to run repeatedly:
    box_t * box;
    camera_fb_t * frame;
    int startx, endx;

    startx = endx = 0;
    for (int i1 = 0; i1++ < 5; ) {
      Serial.printf("Iteration #:%d", i1);

      frame = esp_camera_fb_get(); // capture the image frame from the camera
      if (frame == NULL) {
        Serial.println("frame2 null; esp_camera_fb_get() FAILED!!!");
        setup_camera();
        continue;
      }
      else {
        Serial.println("frame2 NOT NULL; esp_camera_fb_get() PASSED!!!");
      }
```

```
      dl_matrix3du_t *image_matrix = dl_matrix3du_alloc(1, frame->width,
frame->height, 3);  // allocate memory for image analysis
      fmt2rgb888(frame->buf, frame->len, frame->format, image_matrix->item);  //
convert the format to rgb
      esp_camera_fb_return(frame);  // release the frame memory
      box_array_t * boxes = face_detect(image_matrix, &mtmn_config);  // Use the AI
to detect faces and return the (x,y) coordinates

      if (boxes == NULL) { // Found nobody
        Serial.println("AM I ALONE?");
        for (int i2 = 0; i2++ < 3;) {
          digitalWrite(33, LOW);
          delay(100);
          digitalWrite(33, HIGH);
          delay(100);
          //setup_servo(12);
          //sweep_servo(5, 0, 180);
        }
        continue;
      }  // I'm not alone!
      else if (boxes != NULL) {
        detections = detections + 1;
        Serial.printf("Faces detected %d times, %d \n ", detections, boxes->len);
        digitalWrite(33, HIGH);
        int i = 0;
        uint16_t p[4];
        for (i = 0; i < boxes->len; i++) {
          for (int j = 0; j < 4; j++) {
            p[j] = (uint16_t)boxes->box[i].box_p[j];
          }
          Serial.printf("%d %d %d %d \n", p[0], p[1], p[2], p[3]);
        }
        Serial.println("Loop begins! 6");

        //            dl_lib_free(boxes->score);
        //            dl_lib_free(boxes->box);
        //            dl_lib_free(boxes->landmark);
        //            dl_lib_free(boxes);

        if (startx == 0 || (p[0] < startx))
          startx = p[0];
        if (endx == 0 || (p[2] > endx))
          endx = p[2];
      }
      if (image_matrix != NULL) {
        dl_matrix3du_free(image_matrix);
        image_matrix = NULL;
      }
```

```
    }
    Serial.printf("Startx = %d; Endx = %d \n", startx, endx);
    // Switch the start/end because the image is mirrored!
    int endangle = calc_angle(startx);
    int startangle = calc_angle(endx);
    Serial.printf("Startangle = %d; Endangle = %d \n", startangle, endangle);
    if (startangle <= 0 || endangle >= 180 || startangle >= 180 || endangle <= 0) {
      startangle = 0;
      endangle = 180;
    }

    setup_servo(12);
    myservo.write(startangle);
    sweep_servo(10, startangle, endangle);
    // myservo.detach(12);
    // myservo.detach(); // detaches the servo on pin 18 to the servo object
  } // end While
}

/*
   Type: Template function.
   Purpose: None; Necessary to run program because Arduino IDE checks for it. Loop
implemented in void setup.
   Arguments: None.
   Returns: Nothing.
*/
void loop() {
  // Unnecessary since the mainloop is in setup() already
}
```