NYU | TANDON SCHOOL OF ENGINEERING

WINNING THE DRONE RACE

FINAL PROJECT
ADVANCED MECHATRONICS

MITRA VARUN ANAND

SAMANTH

UNDER

Prof VIKRAM KAPILA

INSPIRATION

DEVELOPMENT

- An Augmented Reality based Trainer module to practice accurate control of the drone using Raspberry Pi.
- An obstacle avoiding mechanism using Ping sensors and Arduino Uno to navigate the drone in closed spaces.
- Quick take off mechanism using EZ-Builder and OpenCV.
- Autonomous control of the drone with NodeJS
- Color following drone with EZ Builder.

AUGMENTED REALITY TRAINING MODULE

- Using OpenCV and Raspberry Pi to create a training module for practicing accurate movements.
- Raspberry pi camera continuously tries to track a red marker on top of the drone.
- When the radius of the AR circle matches with radius of circle on top of the drone, the user gets 1 point.

CREATION OF VIRTUAL TRACK

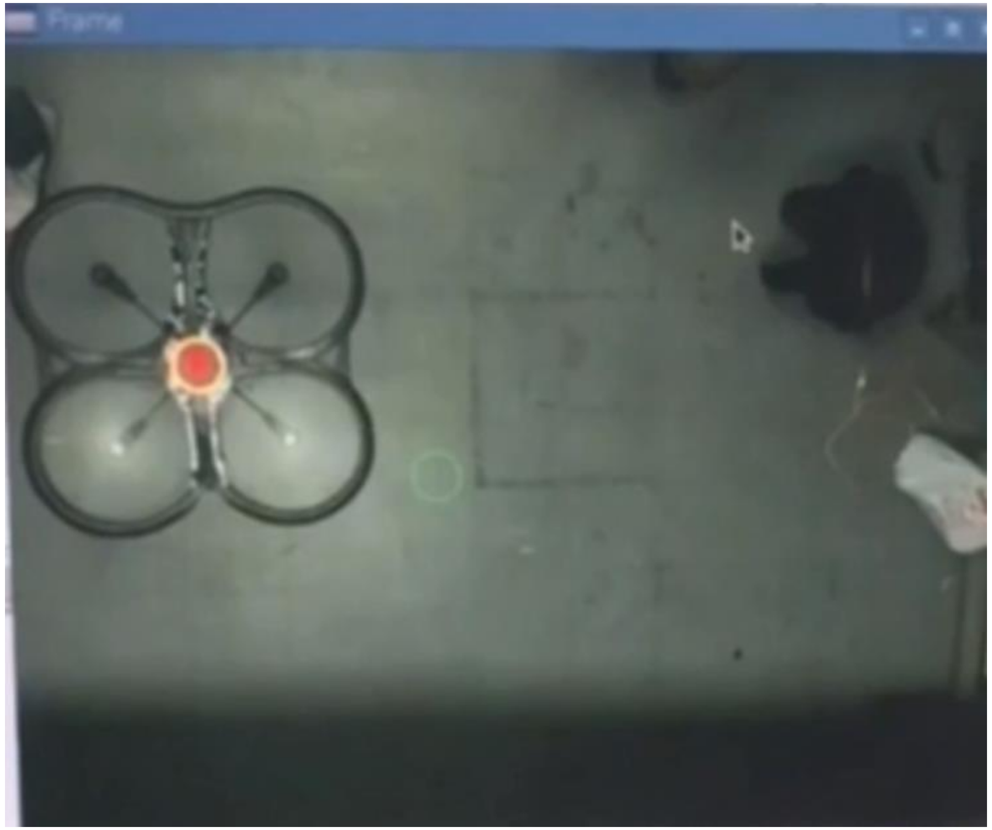
- We used ‘addweighted’ function of openCV to achieve
- cv2.addWeighted(src1, alpha, src2, beta, gamma[, dst[, dtype]]) → dst¶

Parameters:

- src1 – first input array.
- alpha – weight of the first array elements.
- src2 – second input array of the same size and channel number as src1.
- beta – weight of the second array elements.
- dst – output array that has the same size and number of channels as the input arrays.
- gamma – scalar added to each sum.
- dtype – optional depth of the output array; when both input arrays have the same depth, dtype can be set to -1, which will be equivalent to src1.depth().

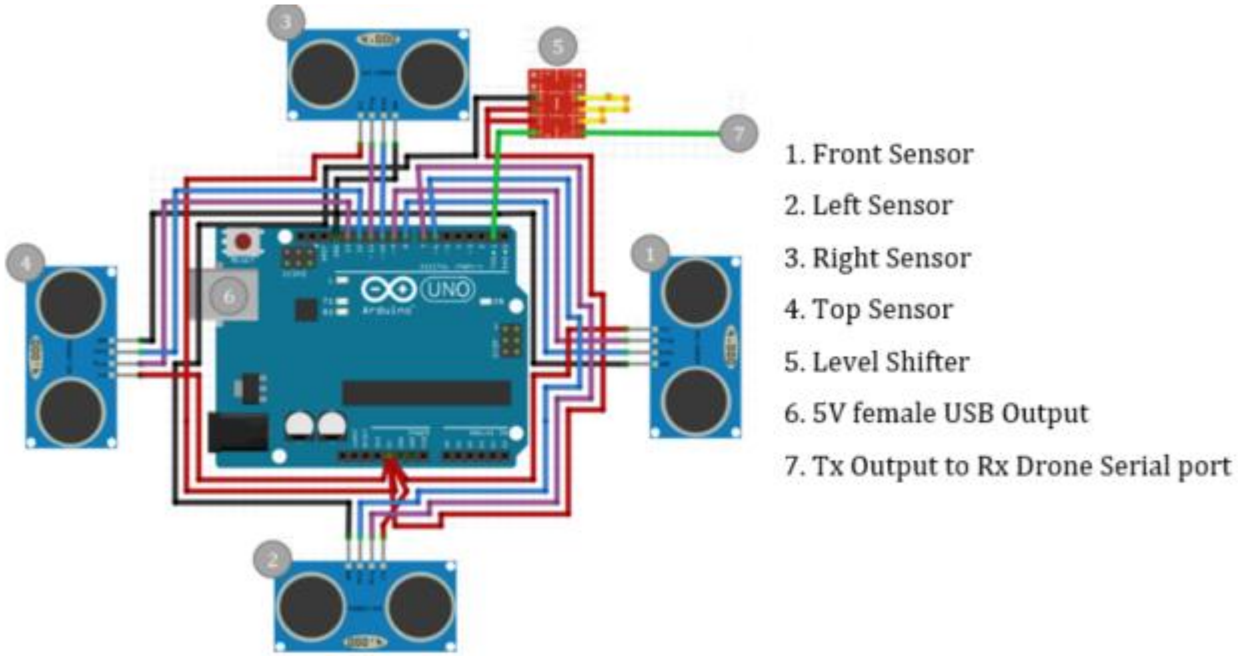
dst = src1*alpha + src2*beta + gamma;

CONTINOUS TRACKING OF DRONE



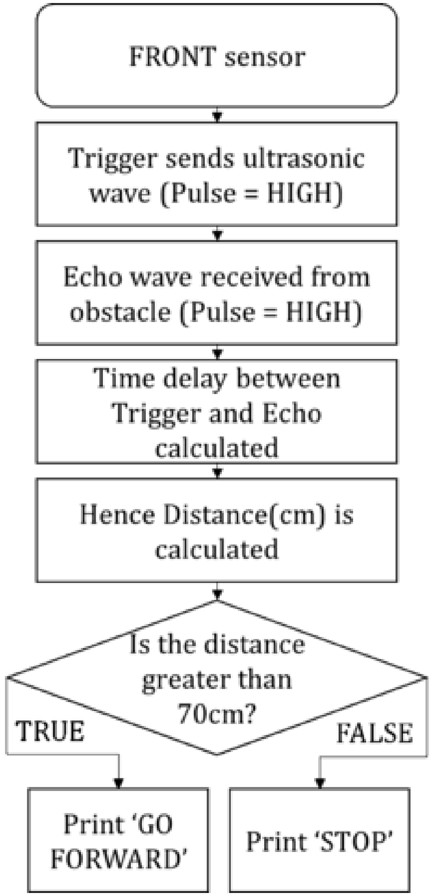
RESULT

COLLISION AVOIDANCE



- Telnet to 192.168.1.1
- Copy the AR Drone node.js file(converted from official AR Drone api)
- Connect the circuit as shown.
- All set to go!

PRINCIPLE



Serial Print	Client API Command	Drone Function
T	client.land( )	LAND
R	client.right(speed)	RIGHT, maintains forward heading
L	client.left(speed)	LEFT, maintains forward heading
F	client.front(speed)	Goes FORWARD
P	client.right(speed)	RIGHT (Front blocked)
Q	client.left(speed)	LEFT (Front blocked)
S	client.stop()	STOP

CODE

```
#define FRONT_TRIG 13 //FRONT
#define FRONT_ECHO 12 //FRONT
#define RIGHT_TRIG 11 //RIGHT
#define RIGHT_ECHO 10 //RIGHT
#define LEFT_TRIG 9 //LEFT
#define LEFT_ECHO 8 //LEFT
#define TOP_TRIG 7 //TOP
#define TOP_ECHO 6 //TOP

void setup() {
  Serial.begin (9600);
  pinMode(FRONT_TRIG, OUTPUT);
  pinMode(FRONT_ECHO, INPUT);
  pinMode(RIGHT_TRIG, OUTPUT);
  pinMode(RIGHT_ECHO, INPUT);
  pinMode(LEFT_TRIG, OUTPUT);
  pinMode(LEFT_ECHO, INPUT);
  pinMode(TOP_TRIG, OUTPUT);
  pinMode(TOP_ECHO, INPUT);
}

void loop() {
    long duration, FRONT, RIGHT, LEFT, TOP; // Duration used to calculate distance of an object from each sensor

  digitalWrite(TOP_TRIG, LOW);            // LOW triggered to ensure no interference from incoming signals, before triggering HIGH
  delayMicroseconds(2);
  digitalWrite(TOP_TRIG, HIGH);           // Send outs ultrasonic wave
  delayMicroseconds(10);                  // Delay allows for ample time to receive the echo signal from object
  digitalWrite(TOP_TRIG, LOW);
  duration = pulseIn(TOP_ECHO, HIGH);     // Calculates time taken to receive signal from reflected signal, pulse is HIGH when signal is received
  TOP = (duration/2) / 29.1;              // Calculates distances using the time calculated above and the speed of sound (300m/s)
  digitalWrite(FRONT_TRIG, LOW);
  delayMicroseconds(2);
  digitalWrite(FRONT_TRIG, HIGH);
  delayMicroseconds(10);
  digitalWrite(FRONT_TRIG, LOW);
  duration = pulseIn(FRONT_ECHO, HIGH);
  FRONT = (duration/2) / 29.1;
  digitalWrite(RIGHT_TRIG, LOW);
  delayMicroseconds(2);
  digitalWrite(RIGHT_TRIG, HIGH);
  delayMicroseconds(10);
  digitalWrite(RIGHT_TRIG, LOW);
  duration = pulseIn(RIGHT_ECHO, HIGH);
```

Done Saving.

2

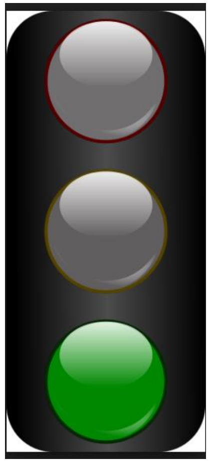
```
duration = pulseIn(RIGHT_ECHO, HIGH);
RIGHT = (duration/2) / 29.1;
digitalWrite(LEFT_TRIG, LOW);
delayMicroseconds(2);
digitalWrite(LEFT_TRIG, HIGH);
delayMicroseconds(10);
digitalWrite(LEFT_TRIG, LOW);
duration = pulseIn(LEFT_ECHO, HIGH);
LEFT = (duration/2) / 29.1;

  if (TOP < 60) {
    Serial.println("T\n");
  }
  else{
  if ((RIGHT < 90) && (FRONT > 80)) {
      Serial.println("L\n");
  }
  if ((LEFT < 90) && (FRONT > 80)) {
      Serial.println("R\n");
  }
  if (FRONT > 90) {
      Serial.println("F\n");
  }
  if (FRONT >= 7 && FRONT <= 50) {
      Serial.println("S\n");
  }
  else{
     if ((LEFT < RIGHT) && (FRONT >=7 && FRONT <=89)) {
      Serial.println("P\n");
      }
     if ((LEFT > RIGHT) && (FRONT >=7 && FRONT <=89)) {
      Serial.println("Q\n");
        }
     }
  }
```

VIDEO

QUICK FIRE TAKEOFF

- Humans are slower to react to a green signal, delaying the take off once the race starts.



- Our mechanism makes use of color detection to immediately start the take off process, better yet, give an initial push to kick-start the race.

CODE

Event EZ-Script Editor

Edit

You are editing an event with EZ-Script. While this code executes, it will block other events of this control from executing.

Ensure you are not performing any time consuming operations in this event code.

Format Code (Alt-F) | Font Smaller | Font Larger | Enable Intellisense | Verbose Debug

```
Define:
num_version_config              = 1
num_version_mb                  = 34
num_version_soft                = 2.4.8
drone_serial                    = PS721801BJ5J106582
soft_build_date                 = 2013-09-16 14:49
motor1_soft                     = 1.43
motor1_hard                     = 6.0
motor1_supplier                 = 1.1
motor2_soft                     = 1.43
motor2_hard                     = 6.0
motor2_supplier                 = 1.1
motor3_soft                     = 1.43
motor3_hard                     = 6.0
motor3_supplier                 = 1.1
motor4_soft                     = 1.43
motor4_hard                     = 6.0
motor4_supplier                 = 1.1
ardrone_name                    = My ARDrone
flying_time                     = 125
navdata_demo                    = TRUE
com_watchdog                    = 2
video_enable                    = TRUE
vision_enable                   = TRUE
vbat_min                        = 9000
gps_soft                        = 0.0
gps_hard                        = 0.0
timezone                        = +0000
battery_type                    = 0
gps_soft_update                 = 4.1.2-P6

[control]
accs_offset                     = { -4.2177085e+03 3.8161274e+03 4.1904790e+03 }
accs_gains                      = {  1.9323235e+00 1.7146780e-01 -5.6314021e-02  1.5902515e-01 -1.9676199e+00 -4.0446233e-02  -7.0570141e-02 -8.04469
gyros_offset                    = { 2.5123751e+01 5.7458752e+01 2.8850000e+00 }
gyros_gains                     = { 1.0595095e-03 -1.0566706e-03 -1.0661810e-03 }
gyros110_offset                 = { 1.6625000e+03 1.6625000e+03 }
gyros110_gains                  = { 1.5271631e-03 -1.5271631e-03 }
magneto_offset                  = { 2.4706065e+02 3.9613686e+01 0.0000000e+00 }
magneto_radius                  = 3.1971252e+02
gyro_offset_thr_x               = 4.0000000e+00
```

2

```
gyro_offset_thr_y                    = 4.0000000e+00
gyro_offset_thr_z                    = 5.0000000e-01
pwm_ref_gyros                        = 500
osctun_value                         = 62
osctun_test                          = TRUE
altitude_max                         = 5000
altitude_min                         = 50
outdoor                              = FALSE
flight_without_shell                 = TRUE
autonomous_flight                    = FALSE
control_level                        = 0
euler_angle_max                      = 2.5000000e-01
control_iphone_tilt                  = 3.4906584e-01
control_vz_max                       = 1.0000000e+03
control_yaw                          = 3.0000000e+00
manual_trim                          = FALSE
indoor_euler_angle_max               = 2.5000000e-01
indoor_control_vz_max                = 1.0000000e+03
indoor_control_yaw                   = 3.0000000e+00
outdoor_euler_angle_max              = 3.4906584e-01
outdoor_control_vz_max               = 1.0000000e+03
outdoor_control_yaw                  = 3.4906585e+00

[network]
ssid_single_player                   = ardrone2_106582
ssid_multi_player                    = ardrone2_106582
wifi_mode                            = 0
owner_mac                            = 00:00:00:00:00:0

[pic]
ultrasound_freq                      = 8
ultrasound_watchdog                  = 3
pic_version                          = 184877090

[video]
camif_fps                            = 30
camif_buffers                        = 2
num_trackers                         = 12
video_on_usb                         = TRUE
video_file_index                     = 1
codec_fps                            = 30
```

3

```
enemy_colors                     = 1
enemy_without_shell              = 0
groundstripe_colors              = 16
detect_type                      = 3
detections_select_h              = 0
detections_select_v_hsync        = 0
detections_select_v              = 0

[syslog]
output                           = 7
max_size                         = 102400
nb_files                         = 5

[custom]
application_desc                 = Default application configuration
profile_desc                     = Default profile configuration
session_desc                     = Default session configuration

[userbox]

[gps]
latitude                         = 5.0000000000000000e+02
longitude                        = 5.0000000000000000e+02
altitude                         = 0.0000000000000000e+00
accuracy                         = 0.0000000000000000e+00

[flightplan]
default_validation_radius        = 3.0000000e+00
default_validation_time          = 1.0000000e-03
max_distance_from_takeoff        = 1000
gcs_ip                           = 3
video_stop_delay                 = 10
low_battery_go_home              = FALSE
automatic_heading                = TRUE
com_lost_action_delay            = 0
altitude_go_home                 = 0.0000000e+00
mavlink_js_roll_left             = x-
mavlink_js_roll_right            = x+
mavlink_js_pitch_front           = y+
mavlink_js_pitch_back            = y-
mavlink_js_yaw_left              = 4
```

4

```
mavlink_js_pitch_back          = y-
mavlink_js_yaw_left            = 4
mavlink_js_yaw_right           = 5
mavlink_js_go_up               = 0
mavlink_js_go_down             = 1
mavlink_js_inc_gains           = 6
mavlink_js_dec_gains           = 7
mavlink_js_select              = 8
mavlink_js_start               = 9

[rescue]

from(ardrone_library.js) load:
arDrone.createClient([ip])
     Takeoff()
     Sleep()
     rollright()
     rollleft()
     Land
Mat image, resized, gray;
 cout<<"Opening Camera..."<<endl;
 if (!Camera.open()) {cerr<<"Error opening the
camera"<<endl;return -1;}
 //set capture properties
 sleep(5);
 Camera.set ( CV_CAP_PROP_FRAME_WIDTH, 1280 );
 Camera.set ( CV_CAP_PROP_FRAME_HEIGHT, 960 );
 Camera.set ( CV_CAP_PROP_BRIGHTNESS, 50);
 Camera.set ( CV_CAP_PROP_CONTRAST, 50);
 namedWindow( "Camera Video", CV_WINDOW_AUTOSIZE );
 Camera.grab();
 Camera.retrieve ( image);
  if (inRange(processed, Scalar(160, 20, 20), Scalar(180, 255, 255),processed));
     drawContours(image, contours, -1, Scalar(255,0,0), 3);
     Takeoff()
     Sleep(1500)
     rollright(1000)
     Land
  else()
  CloseControl()
```

PROCESS

- Connect to ARDrone network through WiFi.
- Open the Script and load on EZBuilder.
- Click connect.
- Whenever the drone detects the color, it will take-off.

VIDEO

Autonomous Control of Drone

- Install node.js .
- Code:

```
arDrone = require('drone');
client  = arDrone.createClient();

client.takeoff();

client
  .after(2000, function() {
    this.up(1);
  /*})
  .after(2000, function() {
    this.animate('turnaround',500);
  })
```

CODE 2

```
.after(5000, function() {
  this.front(1.0);
 })
 .after(2000, function() {
  this.clockwise(0.5);
 })
 .after(5000, function() {
  this.back(0.8);
 })*/
 .after(200, function() {
  this.land();
 });
```

NEEDS IMPROVEMENT/FIXING

- Standalone system incorporating all the features to control with a same controller.
- Fix Green light issue.
- Mods to counter-act forces caused by additional components for stability.
- FPV glasses-stream to make it more similar to the actual race experience.
- A full fledged gaming app incorporating the training module and using custom made Augmented Reality tracks to simulate racing environment.


Thank you