CODING & TESTING

```
#include <SoftwareSerial.h>
SoftwareSerial gsm(8,9);
#include "DHT.h"

#define DHTPIN 2

#define DHTTYPE DHT11 // DHT 11

DHT dht(DHTPIN, DHTTYPE);
#define i1 6
#define i2 5
#define i3 4  #define i4
3 int trigger=A0; int
echo=A1; String a; void
setup() {
Serial.begin(9600);
gsm.begin(9600);
dht.begin();
pinMode(trigger,OUTP
UT);
pinMode(echo,INPUT);
pinMode(i1,OUTPUT);
pinMode(i2,OUTPUT);
pinMode(i3,OUTPUT);
pinMode(i4,OUTPUT);
```

```
} void loop() {   delay(1000); // Wait a few seconds
between measurements    float h = dht.readHumidity();
   // Reading temperature or humidity takes about 250 milliseconds!
float t = dht.readTemperature();
   // Read temperature as Celsius (the default)
float f = dht.readTemperature(true);
   // Read temperature as Fahrenheit (isFahrenheit =
true)    // Check if any reads failed and exit early (to
try again).    if (isnan(h) || isnan(t) || isnan(f) ) {
      Serial.println("Failed  to  read  from  DHT  sensor!");
return;
   }
   Serial.print ("Humidity: ");
   Serial.print (h);
   Serial.print (" %\n");
   Serial.print ("Temperature: ");
   Serial.print (t);
   Serial.print (" *C ");    Serial.print (" %\n");    if(t>36){
gsm.println("AT+CMGF=1");    //Sets the GSM Module in Text
Mode     delay(1000);  // Delay of 1000 milli seconds or 1 second
gsm.println("AT+CMGS=\"+9176391931438   \"\r");
    delay(1000);     gsm.println("Temperature is obnormal");// The SMS
text you want to send     delay(1000);     gsm.println((char)26);// ASCII
code of CTRL+Z     delay(1000);
   }   long distance;   long
duration;
digitalWrite(trigger,HIGH)
```

```
;   delay(10);
digitalWrite(trigger,LOW);
duration=pulseIn(echo,HI
GH);
distance=duration*0.0343/
2;   Serial.println(distance);
while(Serial.available()>0)
{     a=Serial.readString();
    Serial.println(a);
  }
    if(a=="1" && distance>30){
digitalWrite(i4,HIGH);
digitalWrite(i2,HIGH);
digitalWrite(i3,LOW);
digitalWrite(i1,LOW);
    }
    else if(a=="2" && distance>30){
digitalWrite(i2,LOW);
digitalWrite(i4,LOW);
digitalWrite(i3,HIGH);
digitalWrite(i1,HIGH);
    }
    else if(a=="3" && distance>30){
digitalWrite(i2,HIGH);
digitalWrite(i4,LOW);
digitalWrite(i3,LOW);
digitalWrite(i1,LOW);
```

```
    }
    else if(a=="4" && distance>30){
digitalWrite(i4,HIGH);
digitalWrite(i2,LOW);
digitalWrite(i3,LOW);
digitalWrite(i1,LOW);
    }
    else if(a=="0" && distance>30){
digitalWrite(i3,LOW);
digitalWrite(i1,LOW);
digitalWrite(i4,LOW);
digitalWrite(i2,LOW);
    }
  else if(a=="7" && distance<30){
digitalWrite(i3,HIGH);
digitalWrite(i1,HIGH);
digitalWrite(i4,LOW);
digitalWrite(i2,LOW);

    }
  else if(distance<10){
digitalWrite(i3,LOW);
digitalWrite(i1,LOW);
digitalWrite(i4,LOW);
digitalWrite(i2,LOW);
    }
 }
```

CODE FOR ESP32 MODULE

```
#include "esp_camera.h"
#include <WiFi.h>
// WARNING!!! PSRAM IC required for UXGA resolution and high JPEG quality
//            Ensure ESP32 Wrover Module or other board with PSRAM is selected
//            Partial images will be transmitted if image exceeds buffer size
// Select camera model
// #define CAMERA_MODEL_WROVER_KIT // Has PSRAM
//#define CAMERA_MODEL_ESP_EYE // Has PSRAM
//#define CAMERA_MODEL_M5STACK_PSRAM // Has PSRAM
//#define CAMERA_MODEL_M5STACK_V2_PSRAM // M5Camera version B
Has PSRAM
//#define CAMERA_MODEL_M5STACK_WIDE // Has PSRAM
//#define CAMERA_MODEL_M5STACK_ESP32CAM // No PSRAM
#define CAMERA_MODEL_AI_THINKER // Has PSRAM
//#define CAMERA_MODEL_TTGO_T_JOURNAL // No PSRAM
#include "camera_pins.h" const
char* ssid = "iotadmin"; const
char* password = "12345678";
void startCameraServer(); void
setup() {
Serial.begin(115200);
  Serial.setDebugOutput(true);
Serial.println();   camera_config_t config;
config.ledc_channel =
LEDC_CHANNEL_0;
```

```
config…..ledc_timer =
LEDC_TIMER_0;   config.pin_d0 =
Y2_GPIO_NUM;   config.pin_d1 =
Y3_GPIO_NUM;   config.pin_d2 =
Y4_GPIO_NUM;   config.pin_d3 =
Y5_GPIO_NUM;   config.pin_d4 =
Y6_GPIO_NUM;   config.pin_d5 =
Y7_GPIO_NUM;   config.pin_d6 =
Y8_GPIO_NUM;   config.pin_d7 =
Y9_GPIO_NUM;   config.pin_xclk =
XCLK_GPIO_NUM;   config.pin_pclk =
PCLK_GPIO_NUM;   config.pin_vsync
= VSYNC_GPIO_NUM;
config.pin_href = HREF_GPIO_NUM;
config.pin_sscb_sda =
SIOD_GPIO_NUM;
config.pin_sscb_scl =
SIOC_GPIO_NUM;   config.pin_pwdn =
PWDN_GPIO_NUM; config.pin_reset =
RESET_GPIO_NUM;

  config.xclk_freq_hz             =             20000000;
config.pixel_format = PIXFORMAT_JPEG;
  // if PSRAM IC present, init with UXGA resolution and higher JPEG
quality   // for larger pre-allocated frame buffer.
  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)                                   ||
defined(CAMERA_MODEL_M5STACK_ESP32C
AM)   s->set_vflip(s, 1);   s->set_hmirror(s, 1);
#endif
  WiFi.begin(ssid, password);   while
(WiFi.status() != WL_CONNECTED) {
delay(500);     Serial.print(".");
  }
  Serial.println("");
  Serial.println("WiFi                    connected");
startCameraServer ();
  Serial.print("Camera Ready! Use 'http://");
  Serial.print(WiFi.localIP());
  Serial.println("' to connect");
} void
loop() {
// put
your
main
code
here, to
run
repeate
dly:
```

```
delay(1
0000);
}
```