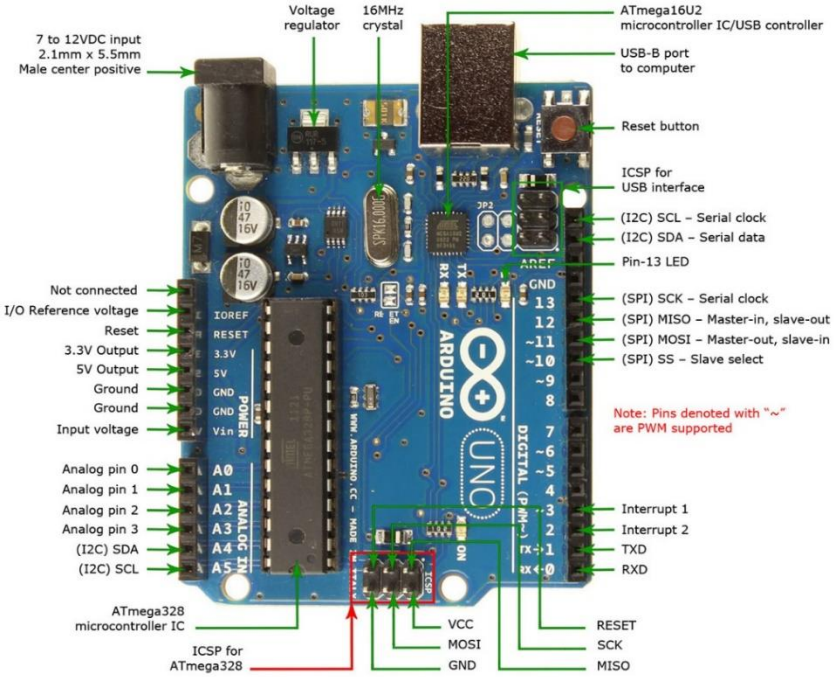
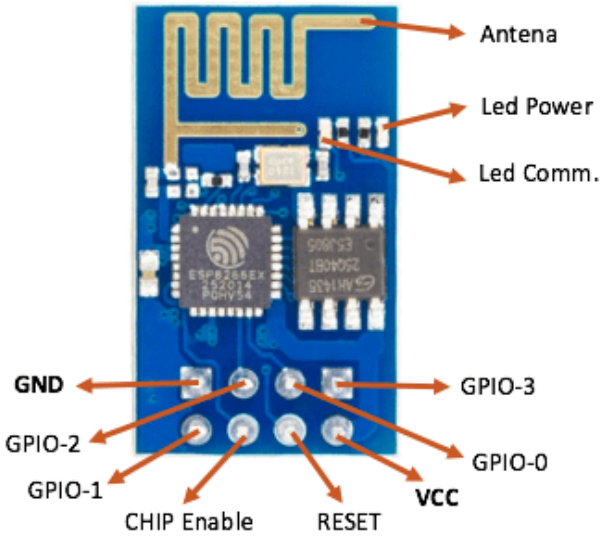
LAMPIRAN

Lampiran 1 Data Sheet Arduino UNO



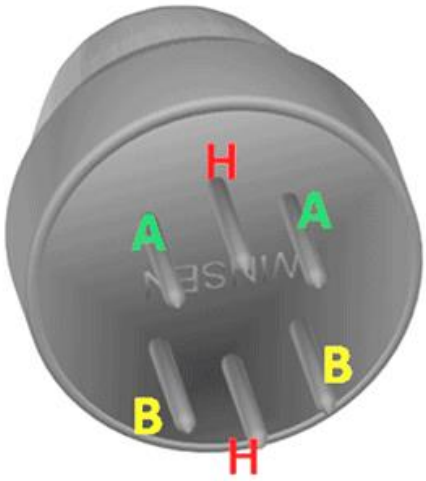
Tegangan Operasi	5V
Tegangan Input	(disarankan) 7—12V
Batas Tegangan Input	6—2OV
Pin Digital I/O	6
Arus DC per I/O Pin	40 mA
Arus DC untuk pin	3.3V 50 mA
Flash Memory	32 KB (ATmega328) , di mana 0,5 KB digunakan oleh bootloader
SRAM	2 KB (Atmega328)
EEPROM	1 KB (Atmega328)
Clock	16 MHz

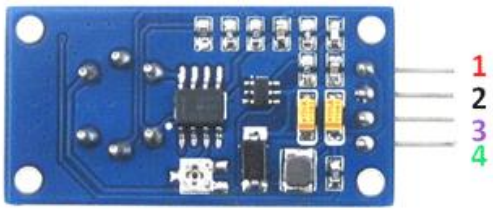
Lampiran 2 Data Sheet ESP-01



- Besar RAM 96 kB, instruction RAM 64 kB
- 32-bit RISC CPU
- External QSPI flash – 512 KiB to 4 MiB
- Tegangan kerja masukan 3.3 Vdc
- Jaringan wifi pada 802.11 b/g/n
- Pada mode 802.11b output power-nya +19.5dBm
- Menggunakan sistem Wi-Fi Direct (P2P), soft-AP
- Power down leakage current of 10uA
- Wake up and transmit packets in < 2ms
- Integrated TCP/IP protocol stack
- Standby power consumption of < 1.0mW (DTIM3)
- SDIO 1.1 / 2.0, SPI, UART
- 10-bit ADC
- Interface : SPI, I²C
- STBC, 11 MIMO, 21 MIMO
- A-MPDU & A-MSDU aggregation & 0.4ms guard interval

Lampiran 3 Data Sheet Sensor MQ2

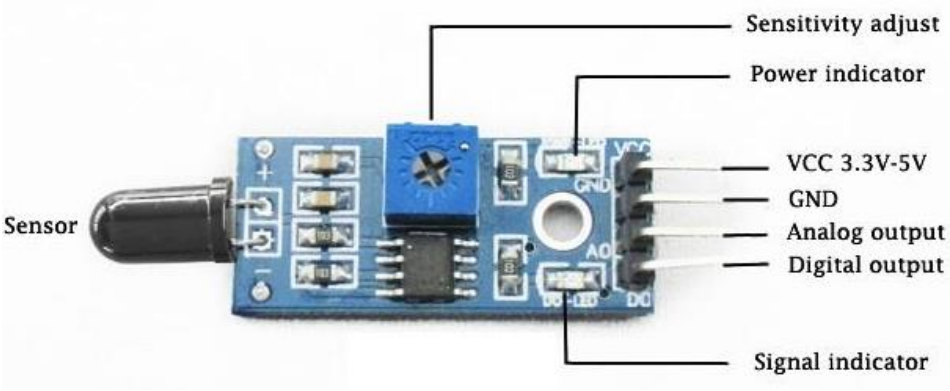




Pin No.	Pin Name
1	Vcc(+5V)
2	Ground
3	Digital Out
4	Analog out

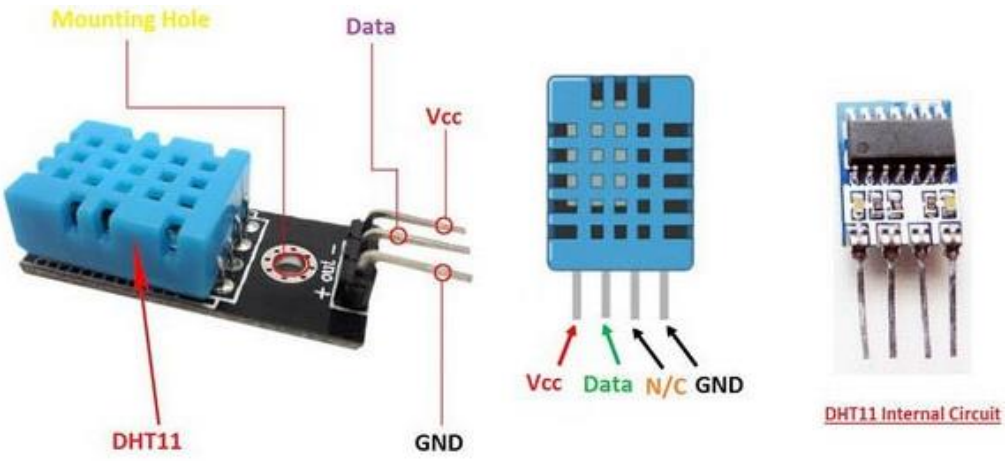
Catu daya pemanas	5V AC/DC
Catu daya rangkaian	5VDC
Range pengukuran	200 – 5000 ppm untuk LPG, propane 300 – 5000 ppm untuk butane 5000 – 20000 ppm untuk methane 300 – 5000 ppm untuk Hidrogen
Keluaran	analog (perubahan tegangan)

Lampiran 4 Data Sheet Sensor Api (KY- 026)



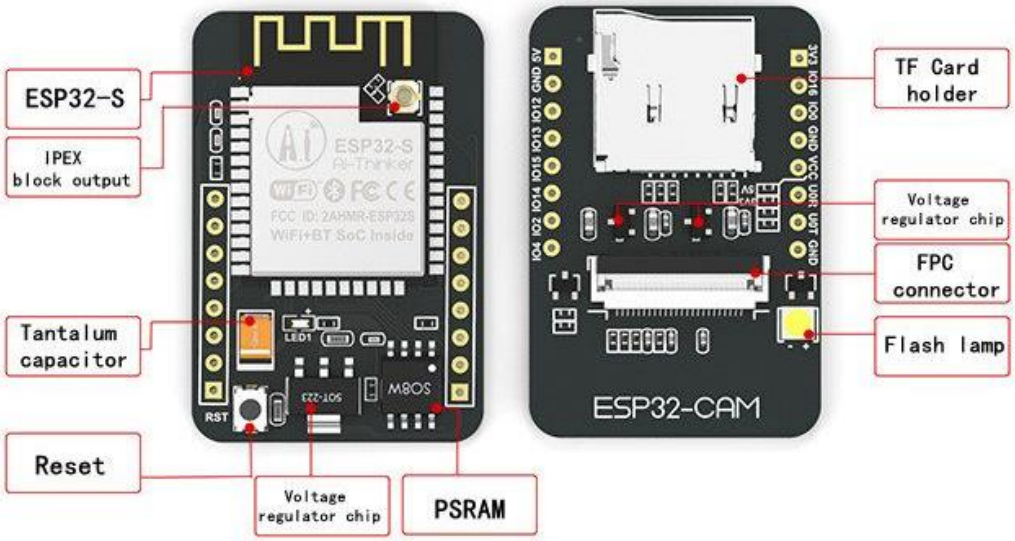
Overview	Specifications	Applications
• Sensitive to flame and radiation • Features wide range voltage comparator LM393 • *Adjustable sensitivity* • Signal output indicator	• Spectrum range : 760 nm ~ 1100 nm (ie. 80CM tested by regular lighter) • Detection angle: -60 - 60 degree • Power: 3.3V ~ 5.3V • Operating temperature: -25°C ~ 85°C • Dimension: 29.2 mm * 11.2mm • Mounting holes size: 2.0mm	• Fire detection • Fire fighting robot • Fire alarm

Lampiran 5 Data Sheet Sensor Suhu DHT 11



Tegangan input	3,5 – 5 VDC
Sistem komunikasi	Serial (single – Wire Two way
Range suhu	0^0C – 50^0C
Range kelembaban	20% – 90% RH
Akurasi	±2^0C (temperature) ±5% RH (humidity)

Lampiran 6 Data Sheet ESP32 Cam



Spesifikasi:

- Low-power dual-core 32-bit CPU for application processors
- Main frequency up to 240MHz, computing power up to 600 DMIPS
- Built-in 520 KB SRAM, external 4M PSRAM
- Supports interfaces such as UART/SPI/I2C/PWM/ADC/DAC
- Support OV2640 and OV7670 cameras, built-in flash
- Support image WiFi upload
- Support TF card
- Support multiple sleep modes
- Embedded Lwip and FreeRTOS
- Support STA/AP/STA+AP working mode
- Support Smart Config/AirKiss one-click distribution network
- Support secondary development

Lampiran 7 Program Keseluruhan Sistem Pendeteksi Kebakaran Multisensor

```
#define BLYNK_PRINT Serial
#include <ESP8266_Lib.h>
#include <BlynkSimpleShieldEsp8266.h>
#include <dht.h>
#include <DHT.h>
#include <Servo.h>
dht DHT;
const int PINBUZZER = 12;

char auth[] = "qjH1zpUBbhGTkcg58WOutexhVpaRWCd_";
char ssid[] = "LAB ELEKTRO";
char pass[] = "elektro@ypkp!";

#define BLYNK_PRINT Serial
#define DHT11_PIN A0

BlynkTimer timer;

#define EspSerial Serial1
#include <SoftwareSerial.h>
SoftwareSerial EspSerial(2,3); // RX, TX

#define ESP8266_BAUD 9600
ESP8266 wifi(&EspSerial);

void setup()
{
  Serial.begin(9600);
  EspSerial.begin(ESP8266_BAUD);
  delay(10);
  Blynk.begin(auth, wifi, ssid, pass);
  Blynk.run();
  pinMode (PINBUZZER, OUTPUT);
}

void loop()
{
 int   H = DHT.humidity;
 int   T = DHT.temperature;
 float A = analogRead(A2);
 float G = analogRead(A3);
 float S = analogRead(A4);

 float VApi  = A*5.00/1024;
 float VGas  = G*5.00/1024;
 float VAsap = S*5.00/1024;

 Blynk.virtualWrite(V1, H);
 Blynk.virtualWrite(V2, T);
 Blynk.virtualWrite(V3, A);
 Blynk.virtualWrite(V4, G);
 Blynk.virtualWrite(V5, S);

/*
// test begin

  // check WiFi connection:
```

```
  if (WiFi.status() != WL_CONNECTED)
  {

    // check delay:
    // if (millis() - lastConnectionAttempt >= connectionDelay)
    {
      lastConnectionAttempt = millis();

      // attempt to connect to Wifi network:
      float status = WiFi.status();
        Serial.println("Lost connections. Trying to
reconnect...");
        Serial.println(status);
        // Blynk.begin(auth, wifi, ssid, pass);

    }
  }
  else
  {
    Blynk.begin(auth, wifi, ssid, pass);
    Blynk.run();
  }
// test end
*/
{
  {
  DHT.read11(DHT11_PIN);
  }
  Serial.println("");
  Serial.print("Api : ");
  Serial.print(A);
  Serial.print(" ppm - ");
  Serial.print(VApi);
  Serial.print(" Volt");
  Serial.println("");
  Serial.print("Gas : ");
  Serial.print(G);
  Serial.print(" ppm - ");
  Serial.print(VGas);
  Serial.print(" Volt");
  Serial.println("");
  Serial.print("Asap : ");
  Serial.print(S);
  Serial.print(" ppm - ");
  Serial.print(VAsap);
  Serial.print(" Volt");
  Serial.println("");
  Serial.print("Humidity : ");
  Serial.print(H);
  Serial.print("%  ");
  Serial.println("");
  Serial.print("temperature : ");
  Serial.print(T);
  Serial.print("C  ");

 Blynk.run();
 delay(500);
}
```

Lampiran 8 Streaming Camera (ESP32-CAM)

```
 /*******************************************************************
 *  Board Settings:
 *  Board: "ESP32 Wrover Module"
 *  Upload Speed: "921600"
 *  Flash Frequency: "80MHz"
 *  Flash Mode: "QIO"
 *  Partition Scheme: "Hue APP (3MB No OTA/1MB SPIFFS)"
 *  Core Debug Level: "None"
 *  COM Port: Depends *On Your System*

*******************************************************************/

#include "src/OV2640.h"
#include <WiFi.h>
#include <WebServer.h>
#include <WiFiClient.h>
#include "src/SimStreamer.h"
#include "src/OV2640Streamer.h"
#include "src/CRtspSession.h"
#define ENABLE_WEBSERVER
#define ENABLE_RTSPSERVER
#ifdef ENABLE_OLED
#include "SSD1306.h"
#define OLED_ADDRESS 0x3c
#define I2C_SDA 14
#define I2C_SCL 13
SSD1306Wire display(OLED_ADDRESS, I2C_SDA, I2C_SCL,
GEOMETRY_128_32);
bool hasDisplay; // untuk pemeriksaan perangkat saat runtime
#endif
#define CAMERA_MODEL_AI_THINKER
#include "camera_pins.h"
OV2640 cam;
#ifdef ENABLE_WEBSERVER
WebServer server(80);
#endif
#ifdef ENABLE_RTSPSERVER
WiFiServer rtspServer(8554);
#endif

#ifdef SOFTAP_MODE
IPAddress apIP = IPAddress(192, 168, 1, 1);
#else

const char *ssid =     "your_ssid";
const char *password = "your_pass";

#endif
#ifdef ENABLE_WEBSERVER
void handle_jpg_stream(void)
{
    WiFiClient client = server.client();
    String response = "HTTP/1.1 200 OK\r\n";
    response += "Content-Type: multipart/x-mixed-replace;
boundary=frame\r\n\r\n";
    server.sendContent(response);
```

```
          while (1)
    {
        cam.run();
 if (!client.connected())
            break;
        response = "--frame\r\n";
        response += "Content-Type: image/jpeg\r\n\r\n";
        server.sendContent(response);

        client.write((char *)cam.getfb(), cam.getSize());
        server.sendContent("\r\n");
        if (!client.connected())
            break;
    }
}
void handle_jpg(void)
{
    WiFiClient client = server.client();
    cam.run();
    if (!client.connected())
    {
        return;
    }
    String response = "HTTP/1.1 200 OK\r\n";
    response += "Content-disposition: inline;
filename=capture.jpg\r\n";
    response += "Content-type: image/jpeg\r\n\r\n";
    server.sendContent(response);
    client.write((char *)cam.getfb(), cam.getSize());
}
void handleNotFound()
{
    String message = "Server is running!\n\n";
    message += "URI: ";
    message += server.uri();
    message += "\nMethod: ";
    message += (server.method() == HTTP_GET) ? "GET" : "POST";
    message += "\nArguments: ";
    message += server.args();
    message += "\n";
    server.send(200, "text/plain", message);
}
#endif
void lcdMessage(String msg)
{
  #ifdef ENABLE_OLED
    if(hasDisplay) {
        display.clear();
        display.drawString(128 / 2, 32 / 2, msg);
        display.display();
    }
  #endif
}
void setup()
{
  #ifdef ENABLE_OLED
    hasDisplay = display.init();
```

```
    if(hasDisplay) {
        display.flipScreenVertically();
        display.setFont(ArialMT_Plain_16);
        display.setTextAlignment(TEXT_ALIGN_CENTER);
    }


    #endif
    lcdMessage("booting");
    Serial.begin(115200);
    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;
    config.frame_size = FRAMESIZE_SVGA;
    config.jpeg_quality = 12;
    config.fb_count = 2;

    #if defined(CAMERA_MODEL_ESP_EYE)
      pinMode(13, INPUT_PULLUP);
      pinMode(14, INPUT_PULLUP);
    #endif
    cam.init(config);
    IPAddress ip;
#ifdef SOFTAP_MODE
    const char *hostname = "devcam";
   // WiFi.hostname(hostname); // FIXME - find out why undefined
    lcdMessage("starting softAP");
    WiFi.mode(WIFI_AP);
    bool result = WiFi.softAP(hostname, "12345678", 1, 0);
    delay(2000);
    WiFi.softAPConfig(apIP, apIP, IPAddress(255, 255, 255, 0));
    if (!result)
    {
        Serial.println("AP Config failed.");
        return;
    }
    else
    {
        Serial.println("AP Config Success.");
        Serial.print("AP MAC: ");
        Serial.println(WiFi.softAPmacAddress());
```

```
        ip = WiFi.softAPIP();
        Serial.print("Stream Link: rtsp://");
        Serial.print(ip);
        Serial.println(":8554/mjpeg/1");
    }
#else
    lcdMessage(String("join ") + ssid);
    WiFi.mode(WIFI_STA);
    WiFi.begin(ssid, password);
    while (WiFi.status() != WL_CONNECTED)
    {
delay(500);
        Serial.print(F("."));
    }
    ip = WiFi.localIP();
    Serial.println(F("WiFi connected"));
    Serial.println("");
    Serial.println(ip);
    Serial.print("Stream Link: rtsp://");
    Serial.print(ip);
    Serial.println(":8554/mjpeg/1");
#endif
    lcdMessage(ip.toString());
#ifdef ENABLE_WEBSERVER
    server.on("/", HTTP_GET, handle_jpg_stream);
    server.on("/jpg", HTTP_GET, handle_jpg);
    server.onNotFound(handleNotFound);
    server.begin();
#endif
#ifdef ENABLE_RTSPSERVER
    rtspServer.begin();
#endif
}
CStreamer *streamer;
CRtspSession *session;
WiFiClient client; // FIXME, support multiple clients
void loop()
{
#ifdef ENABLE_WEBSERVER
    server.handleClient();
#endif

#ifdef ENABLE_RTSPSERVER
    uint32_t msecPerFrame = 100;
    static uint32_t lastimage = millis();
    // If we have an active client connection, just service that
until gone
    // (FIXME - support multiple simultaneous clients)
    if(session) {
        session->handleRequests(0); // we don't use a timeout
here,
        // instead we send only if we have new enough frames
        uint32_t now = millis();
        if(now > lastimage + msecPerFrame || now < lastimage) { //
handle clock rollover
            session->broadcastCurrentFrame(now);
            lastimage = now;
            // check if we are overrunning our max frame rate
            now = millis();
```

```
            if(now > lastimage + msecPerFrame)
                printf("warning exceeding max frame rate of %d
ms\n", now - lastimage);
        }
        if(session->m_stopped) {
            delete session;
            delete streamer;
            session = NULL;
            streamer = NULL;
        }
    }




 else {
        client = rtspServer.accept();

        if(client) {

            streamer = new OV2640Streamer(&client, cam); // our
streamer for UDP/TCP based RTP transport
            session = new CRtspSession(&client, streamer); // our
threads RTSP session and state
        }
    }
#endif
}
```