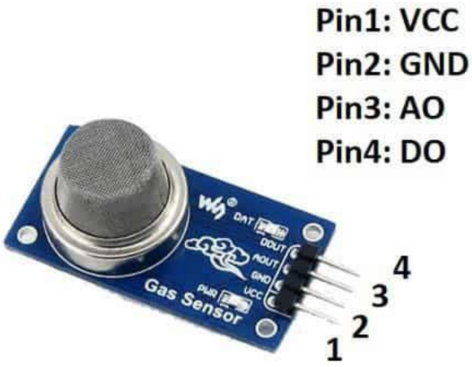
RTisT Project on ESP32

- ## ESP32 INTRODUCTION:

  The ESP32 is a versatile, low-cost microcontroller with built-in Wi-Fi and Bluetooth capabilities. Developed by Espressif Systems, it features a dual-core processor, extensive GPIO, ADC, DAC, and PWM support, making it ideal for IoT, smart home, and wearable applications. Its low power consumption, rich development ecosystem, and compatibility with various programming environments, including Arduino IDE and MicroPython, make it a popular choice among hobbyists and professionals for embedded system projects.

1. Gas Sensor with ESP32:

It detects the concentration of gases in a particular area in which sensor has been deployed.



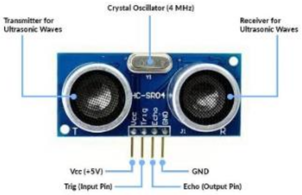
Code:

```
int gaspin = 4
void setup() {
  // put your setup code here, to run once:
  Serial.begin(115200);
  pinMode(gaspin,INPUT);

}

void loop() {
  // put your main code here, to run repeatedly:
  val = analogRead(gaspin);
  Serial.println(val);
}
```

Code is just getting the value of concentration in analog data(in range of 0 to 4095) and printing the value in serial monitor.
If we want to perform some action, we can write the code accordingly.

2. Ultrasonic Sensor:

An ultrasonic sensor is a device that uses high-frequency sound waves to measure the distance to an object. It emits ultrasonic waves and measures the time it takes for the waves to bounce back from the target.


Crystal Oscillator (4 MHz)
Transmitter for Ultrasonic Waves
Receiver for Ultrasonic Waves
HC-SR04
Vcc
Trig
Echo
GND
T
R
Y1
J1
Vcc (+5V)
Trig (Input Pin)
GND
Echo (Output Pin)

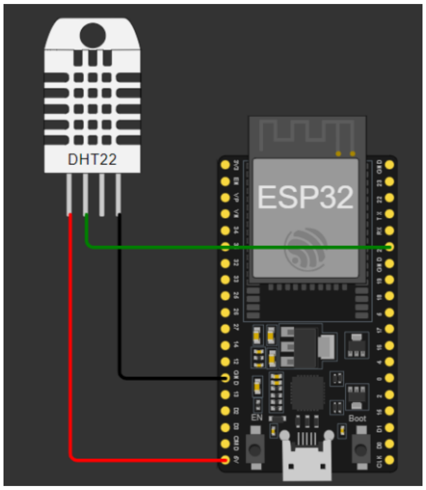
Code:

```
int trigpin = 12;
int echopin = 14;
void setup() {
  // put your setup code here, to run once:
  Serial.begin(115200);
  pinMode(trigpin, OUTPUT);
  pinMode(echopin, INPUT);
}

void loop() {
  // put your main code here, to run repeatedly:
  digitalWrite(trigpin, LOW);
  delay(10);
  digitalWrite(trigpin, HIGH);
  delay(10);
  digitalWrite(trigpin, LOW);
  int time = pulseIn(echopin,HIGH);
  delay(1000);
  Serial.println(time);
}
```

Code is just getting the time between two echoes, returned to module. Module will send ultrasonic sound waves and and will get the that wave again and ESP32 will calculate the time accordingly.

3. Temperature and Humidity Sensor:

A temperature and humidity sensor is a device that measures the ambient temperature and relative humidity in the environment. These sensors often combine both functions in a single module, providing crucial data for climate control, weather monitoring, and environmental assessment.



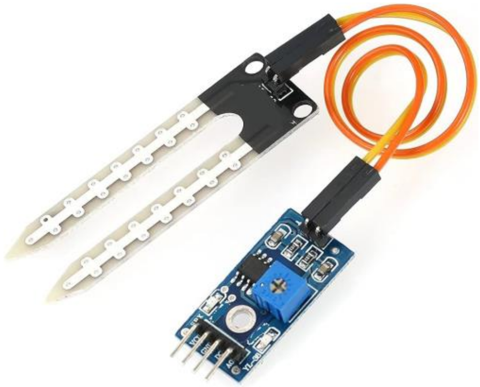
Code:

```
#include <DHT.h>
DHT HT(21,DHT22);
void setup() {
  // put your setup code here, to run once:
  Serial.begin(115200);
  HT.begin();
}

void loop() {
  // put your main code here, to run repeatedly:
  float Humidity = HT.readHumidity();
  Serial.print("Humidity is:");
  Serial.println(Humidity);
  float Temp = HT.readTemperature();
  Serial.print("Temperature is:");
  Serial.println(Temp);
  delay(1000);
}
```

We include DHT library and made a object. Code just explains how to read Temperature and Humidity and print on the serial monitor.

4. Soil Moisture Sensor:

A soil moisture sensor is a device that measures the water content in soil. It helps in determining the moisture level, which is critical for agriculture, gardening, and environmental monitoring.



Code:

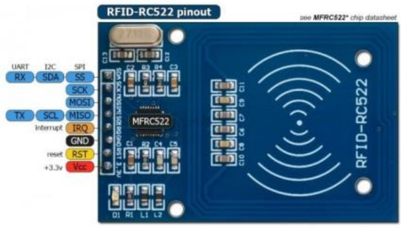
```
int data = 16;
void setup(){
  pinMode(data,INPUT);
  Serial.begin(115200);
}
void loop(){
  int val = analogRead(data);
  Serial.println(val);
  delay(100);
}
```

Code will get the input value from the sensor. In more moisture, it will return high value and in less moisture, will return low.

Code will just print the value in Serial monitor. We can do further operations using the value.

5. <u>RFID Module:</u>

RFID, or Radio Frequency Identification, is a technology that uses electromagnetic fields to automatically identify and track tags attached to objects. An RFID module typically consists of three main components:1. RFID tag, 2. RFID reader, 3. Antenna



Code:

```
#include <SPI.h>
#include <MFRC522.h>
#define RST_PIN 9
#define SS_PIN  10

MFRC522 mfrc522(SS_PIN, RST_PIN);
void setup() {
  Serial.begin(115200);
  while (!Serial);
  SPI.begin();
  mfrc522.PCD_Init();
  delay(4);
  mfrc522.PCD_DumpVersionToSerial();
  Serial.println(F("Scan PICC to see UID, SAK, type, and data
blocks..."));
}
```

```
void loop() {
  if ( ! mfrc522.PICC_IsNewCardPresent()) {
    return;
  }

  if ( ! mfrc522.PICC_ReadCardSerial()) {
    return;
  }
  mfrc522.PICC_DumpToSerial(&(mfrc522.uid));
}
```

➔ This code is to find the UID of any tag. We started our SPI bus and made MFRC522 object. In setup() code, we initialize SPI and mfrc522. In loop() code, we read the UID using PICC_ReadCardSerial() and print it in serial monitor using PICC_DumpToSerial().

```
#include <SPI.h>
#include <MFRC522.h>
#define SS_PIN 10
#define RST_PIN 9
MFRC522 mfrc522(SS_PIN, RST_PIN);
void setup()
{
  Serial.begin(115200);   // Initiate a serial communication
  SPI.begin();      // Initiate  SPI bus
  mfrc522.PCD_Init();   // Initiate MFRC522
```

```
 Serial.println("Approximate your card to the reader...");
 Serial.println();


}
void loop()
{
 // Look for new cards
 if ( ! mfrc522.PICC_IsNewCardPresent())
 {
   return;
 }
 // Select one of the cards
 if ( ! mfrc522.PICC_ReadCardSerial())
 {
   return;
 }
 //Show UID on serial monitor
 Serial.print("UID tag :");
 String content= "";
 byte letter;
 for (byte i = 0; i < mfrc522.uid.size; i++)
 {
    Serial.print(mfrc522.uid.uidByte[i] < 0x10 ? " 0" : " ");
    Serial.print(mfrc522.uid.uidByte[i], HEX);
    content.concat(String(mfrc522.uid.uidByte[i] < 0x10 ? " 0" : " "));
    content.concat(String(mfrc522.uid.uidByte[i], HEX));
  }
  Serial.println();
  Serial.print("Message : ");
  content.toUpperCase();
  if (content.substring(1) == "F9 D1 F3 6E") //change here the UID of the card/cards that you want to give access
  {
    Serial.println("Authorized access");
    Serial.println();
    delay(3000);
  }
  else
  {
    Serial.println(" Access denied");
    Serial.println();
    delay(3000);
  }
}
```

➔ This code is for checking the card UID. In setup() code, we start SPI bus and MFRC522. In loop() code, we will see if there is any card, if there is then what is the UID for it and if it matches we will allow to do operation and if not then don’t allow.

6. ESP32 Cam:

The ESP32-CAM is a low-cost, compact development board that integrates the ESP32-S microcontroller with a camera module. It's designed for applications in IoT (Internet of Things), smart home automation, and surveillance, among others.

Code:

```
#include "esp_camera.h"
#include <WiFi.h>
#define CAMERA_MODEL_ESP_EYE
#include "camera_pins.h"

// ===========================
// Enter your WiFi credentials
// ===========================
const char *ssid = "**********";
const char *password = "**********";

void startCameraServer();
void setupLedFlash(int pin);

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_sccb_sda = SIOD_GPIO_NUM;
  config.pin_sccb_scl = SIOC_GPIO_NUM;
  config.pin_pwdn = PWDN_GPIO_NUM;
  config.pin_reset = RESET_GPIO_NUM;
  config.xclk_freq_hz = 20000000;
  config.frame_size = FRAMESIZE_UXGA;
  config.pixel_format = PIXFORMAT_JPEG;  // for streaming
  //config.pixel_format = PIXFORMAT_RGB565; // for face
detection/recognition
  config.grab_mode = CAMERA_GRAB_WHEN_EMPTY;
  config.fb_location = CAMERA_FB_IN_PSRAM;
  config.jpeg_quality = 12;
  config.fb_count = 1;
  // if PSRAM IC present, init with UXGA resolution and higher
JPEG quality
  //                      for larger pre-allocated frame buffer.
  if (config.pixel_format == PIXFORMAT_JPEG) {
    if (psramFound()) {
      config.jpeg_quality = 10;
      config.fb_count = 2;
      config.grab_mode = CAMERA_GRAB_LATEST;
    } else {
      // Limit the frame size when PSRAM is not available
      config.frame_size = FRAMESIZE_SVGA;
      config.fb_location = CAMERA_FB_IN_DRAM;
    }
  } else {
    // Best option for face detection/recognition
    config.frame_size = FRAMESIZE_240X240;
#if CONFIG_IDF_TARGET_ESP32S3
    config.fb_count = 2;
#endif
  }

#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
  if (config.pixel_format == PIXFORMAT_JPEG) {
    s->set_framesize(s, FRAMESIZE_QVGA);
  }

#if defined(CAMERA_MODEL_M5STACK_WIDE) ||
defined(CAMERA_MODEL_M5STACK_ESP32CAM)
  s->set_vflip(s, 1);
  s->set_hmirror(s, 1);
#endif

#if defined(CAMERA_MODEL_ESP32S3_EYE)
  s->set_vflip(s, 1);
#endif

// Setup LED FLash if LED pin is defined in camera_pins.h
#if defined(LED_GPIO_NUM)
  setupLedFlash(LED_GPIO_NUM);
#endif

  WiFi.begin(ssid, password);
  WiFi.setSleep(false);

  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() {
  // Do nothing. Everything is done in another task by the web server
  delay(10000);
}
```

➔ This code will connect the ESP32 cam module with WiFi and will provide an IP address. On browsing that IP address, we can see our camera’s video streaming.

7. Connect ESP32 with Google voice assistant:

We can connect ESP32 with the Blynk IoT. After that, we will configure it with Google voice assistant using IFTTT. Here is the code to connect ESP32 with Blynk Cloud.

Code:

```
#define BLYNK_TEMPLATE_ID "TMPL3LFPBfXgB"
#define BLYNK_TEMPLATE_NAME "Google Assistant"
#define BLYNK_AUTH_TOKEN "uiixYzSUyzA7__BrbIbCJmQ2COBp-ybT"
```

```
#define BLYNK_PRINT Serial

#include <WiFi.h>
#include <WiFiClient.h>
#include <BlynkSimpleEsp32.h>

BlynkTimer timer;

char auth[]=BLYNK_AUTH_TOKEN;
char ssid[] = "Wokwi-Guest";
char pass[] = "";

void setup() {
  Serial.begin(9600);
  Blynk.begin(auth,ssid,pass);
}

void loop() {
  Blynk.run();
  timer.run();
}
```