

# LAPORAN PRAKTIKUM INTERNET OF THINGS (IoT)

Fakultas Vokasi, Universitas Brawijaya

## Praktik Real Hardware ESP 32

*M Bimo Amarulloh*

Fakultas Vokasi, Universitas Brawijaya

Email: bimoamar@gmail.com

### Abstrak

Praktik ini bertujuan untuk mengimplementasikan sistem monitoring suhu dan jarak berbasis mikrokontroler ESP32 dengan perangkat keras nyata. Sistem ini menggunakan dua buah LED sebagai indikator, sensor suhu dan kelembaban DHT22 untuk mengukur kondisi lingkungan, serta sensor ultrasonik HC-SR04 untuk mengukur jarak objek. ESP32 dipilih karena memiliki konektivitas Wi-Fi serta kemampuan pemrosesan yang tinggi untuk aplikasi IoT. LED pertama digunakan sebagai indikator suhu, yang akan menyala jika suhu melebihi ambang batas tertentu. LED kedua berfungsi sebagai indikator jarak, menyala saat objek berada pada jarak yang telah ditentukan. Data dari sensor dibaca secara real-time dan diproses oleh ESP32, kemudian ditampilkan melalui antarmuka serial. Praktik ini memperkenalkan mahasiswa pada pemrograman mikrokontroler, pemrosesan data sensor, serta integrasi komponen elektronika dalam sistem tertanam. Hasil pengujian menunjukkan sistem dapat bekerja secara stabil dan responsif terhadap perubahan suhu dan jarak di lingkungan sekitar. Praktik ini diharapkan menjadi dasar untuk pengembangan sistem monitoring berbasis IoT yang lebih kompleks di masa depan.

**Kata Kunci:** ESP32, DHT22, HC-SR04, LED indikator, sistem monitoring

### Abstract

This project aims to implement a real hardware-based monitoring system using the ESP32 microcontroller. The system integrates two LEDs as indicators, a DHT22 sensor for temperature and humidity measurement, and an HC-SR04 ultrasonic sensor to detect object distance. The ESP32 is selected for its built-in Wi-Fi capability and powerful processing suitable for IoT applications. The first LED serves as a temperature indicator, turning on when the temperature exceeds a predefined threshold. The second LED acts as a distance indicator, lighting up when an object is detected within a certain range. Sensor data is read in real-time and processed by the ESP32, with output displayed through the serial monitor. This practice introduces students to microcontroller programming, sensor data processing, and hardware integration in embedded systems. Testing results show that the system operates reliably and responds well to changes in temperature and distance. This project serves as a foundational step toward developing more advanced IoT-based monitoring systems in the future.

**Keywords:** ESP32, DHT22, HC-SR04, LED indicator, monitoring system

## 1. Introduction (Pendahuluan)

### 1.1 Latar belakang

Perkembangan teknologi mikrokontroler dan Internet of Things (IoT) telah membuka peluang luas dalam pengembangan sistem monitoring cerdas yang efisien dan terjangkau. Salah satu perangkat yang banyak digunakan dalam berbagai aplikasi IoT adalah ESP32, sebuah mikrokontroler yang dilengkapi dengan konektivitas Wi-Fi dan Bluetooth, serta memiliki kemampuan pemrosesan yang tinggi. ESP32 sangat cocok untuk mengintegrasikan berbagai sensor dan aktuator dalam satu sistem terpadu. Dalam praktik ini, digunakan ESP32 sebagai otak utama sistem yang mengontrol dan memproses data dari dua jenis sensor, yaitu sensor suhu dan kelembaban DHT22 serta sensor jarak ultrasonik HC-SR04. DHT22 digunakan untuk mengukur suhu dan kelembaban lingkungan secara real-time, sedangkan HC-SR04 digunakan untuk mendeteksi jarak suatu objek dari sensor. Selain itu, dua buah LED digunakan sebagai indikator sederhana untuk memberikan respon visual terhadap perubahan suhu dan jarak yang terdeteksi. Praktik ini bertujuan untuk memberikan pemahaman langsung kepada mahasiswa tentang bagaimana membangun sistem monitoring berbasis mikrokontroler, mulai dari membaca data sensor, memproses logika kontrol, hingga mengaktifkan aktuator sederhana. Diharapkan dari praktik ini, mahasiswa mampu mengembangkan keterampilan dasar yang diperlukan dalam membangun aplikasi IoT skala kecil maupun besar.

### 1.2 Tujuan eksperimen

1. **Mengenal dan memahami** penggunaan mikrokontroler ESP32 sebagai pusat kendali dalam sistem monitoring berbasis hardware.
2. **Mengintegrasikan** sensor DHT22 untuk membaca data suhu dan kelembaban serta sensor HC-SR04 untuk mengukur jarak objek secara real-time.
3. **Mengimplementasikan logika kontrol** sederhana menggunakan dua buah LED sebagai indikator perubahan nilai suhu dan jarak.
4. **Melatih kemampuan praktis** mahasiswa dalam pemrograman mikrokontroler, pengolahan data sensor, dan pengendalian aktuator dasar.
5. **Membangun dasar pengetahuan** untuk pengembangan sistem monitoring berbasis IoT yang lebih kompleks di masa depan.

## 2. Methodology (Metodologi)

### 2.1 Tools & Materials (Alat dan Bahan)

#### Alat

1. Komputer/Laptop – Digunakan untuk menulis, mengedit, dan mengunggah program ke mikrokontroler.
2. Arduino IDE / VS Code (dengan ekstensi PlatformIO) – Digunakan sebagai lingkungan pengembangan untuk menulis kode program ESP32.
3. Kabel USB – Untuk menghubungkan ESP32 dengan komputer saat pemrograman dan pemantauan serial.

#### Bahan (Komponen Fisik)

1. ESP32 Dev Board – Mikrokontroler utama yang digunakan sebagai pusat kendali sistem.
2. LED (2 buah) – Digunakan sebagai indikator visual untuk suhu dan jarak.
3. Sensor DHT22 – Sensor suhu dan kelembaban untuk membaca kondisi lingkungan.
4. Sensor Ultrasonik HC-SR04 – Digunakan untuk mengukur jarak objek.

5. Breadboard – Papan tempat merangkai komponen elektronik tanpa penyolderan.
6. Kabel Jumper – Untuk menghubungkan ESP32 dengan sensor dan LED di breadboard.

## 2.2 Implementation Steps (Langkah Implementasi)

Berikut adalah langkah-langkah implementasi dalam pembuatan sistem lampu lalu lintas menggunakan simulator Wokwi dan mikrokontroler ESP32:

1. **Persiapkan Peralatan**

Siapkan ESP32, LED, breadboard, dan kabel jumper di atas meja .

2. **Rangkaian LED**

Hubungkan **kaki panjang (anoda)** LED ke salah satu pin digital ESP32, misalnya **GPIO 25**.

Hubungkan **kaki pendek (katoda)** LED ke **GND (Ground)** ESP32.

3. **Koneksikan ESP32 ke Komputer**

Hubungkan ESP32 ke komputer menggunakan kabel USB.

4. **Buka Arduino IDE / VS Code**

Pilih board: *ESP32 Dev Module*

Pilih port COM yang sesuai dengan ESP32

5. **Tulis dan Unggah Program**

Berikut contoh kode sederhana untuk menyalakan dan mematikan LED secara berkala:

```
#include <Arduino.h> // Wajib untuk PlatformIO + ESP32
```

```
// Deklarasi pin LED
```

```
int lampu = 25;
```

```
int lampu2 = 33;
```

```
void setup()
```

```
{
```

```
  Serial.begin(115200); // Inisialisasi komunikasi Serial
```

```
  Serial.println("ESP32 Blinking LED");
```

```
  // Atur pin sebagai OUTPUT
```

```
  pinMode(lampu, OUTPUT);
```

```
  pinMode(lampu2, OUTPUT);
```

```
}
```

```
void loop()
```

```
{
```

```
// Nyalakan kedua LED
digitalWrite(lampu, HIGH);
digitalWrite(lampu2, HIGH);
Serial.println("LED ON");

delay(1000); // Tunggu 1 detik

// Matikan kedua LED
digitalWrite(lampu, LOW);
digitalWrite(lampu2, LOW);
Serial.println("LED OFF");

delay(1000); // Tunggu 1 detik sebelum mengulang
}
```

6. **Unggah Program ke ESP32**

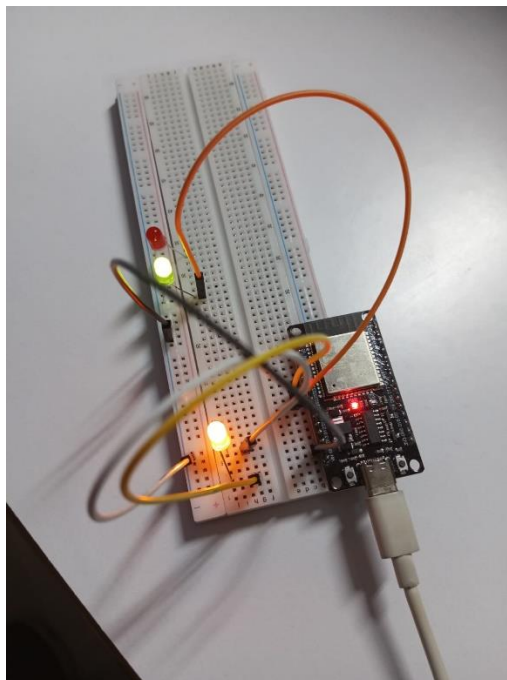
Klik tombol **Upload** dan tunggu hingga proses selesai.

7. **Amati LED**

Jika rangkaian benar, LED akan menyala bersamaan selama 1 detik.

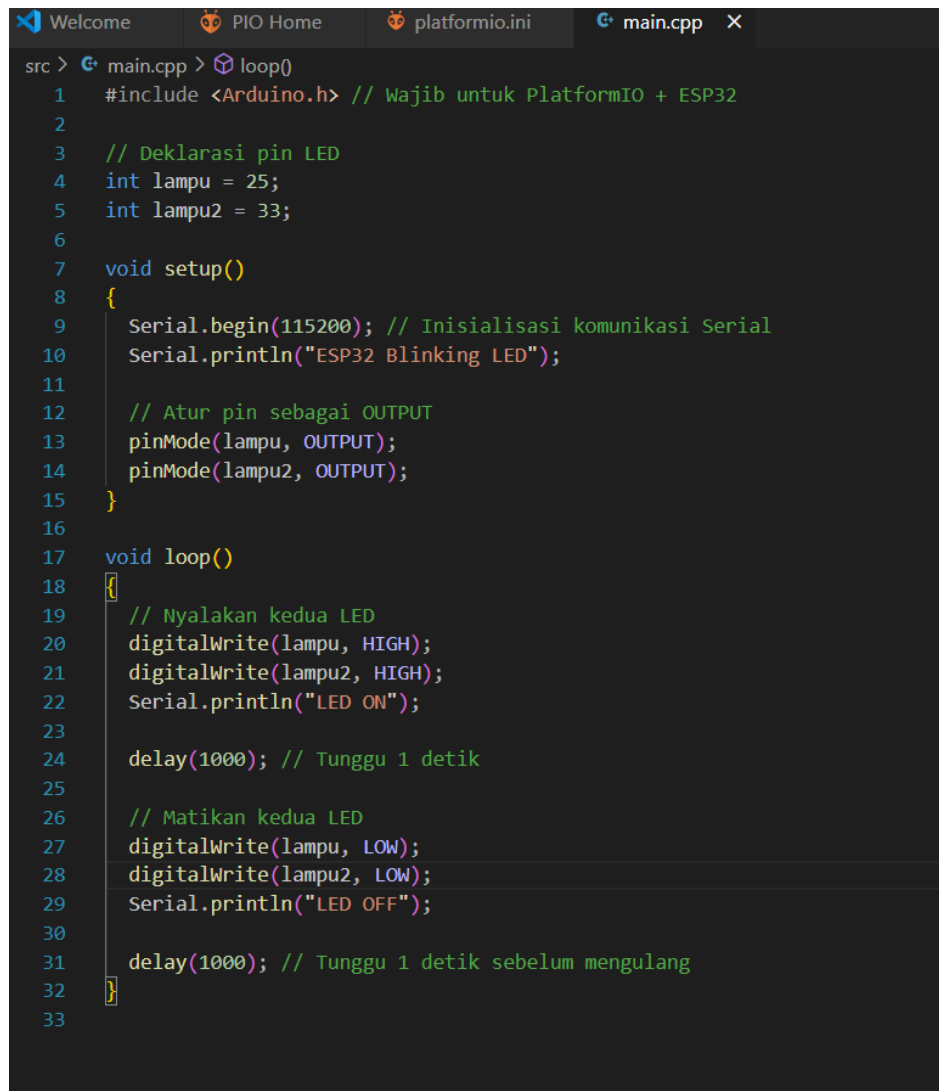
### 3. Results and Discussion (Hasil dan Pembahasan)

#### 3.1 Experimental Results (Hasil Eksperimen)



Kode berjalan dengan baik sesuai dengan keinginan peneliti dimana lampu akan menyala bersamaan selama 1 detik.

#### 4. Appendix (Lampiran)



```
src > main.cpp > loop()
1  #include <Arduino.h> // Wajib untuk PlatformIO + ESP32
2
3  // Deklarasi pin LED
4  int lampu = 25;
5  int lampu2 = 33;
6
7  void setup()
8  {
9      Serial.begin(115200); // Inisialisasi komunikasi Serial
10     Serial.println("ESP32 Blinking LED");
11
12     // Atur pin sebagai OUTPUT
13     pinMode(lampu, OUTPUT);
14     pinMode(lampu2, OUTPUT);
15 }
16
17 void loop()
18 {
19     // Nyalakan kedua LED
20     digitalWrite(lampu, HIGH);
21     digitalWrite(lampu2, HIGH);
22     Serial.println("LED ON");
23
24     delay(1000); // Tunggu 1 detik
25
26     // Matikan kedua LED
27     digitalWrite(lampu, LOW);
28     digitalWrite(lampu2, LOW);
29     Serial.println("LED OFF");
30
31     delay(1000); // Tunggu 1 detik sebelum mengulang
32 }
33
```

Gambar 1. 1 Syntax LED

## 2.2 Langkah Implementasi

### 1. Persiapkan Peralatan

Siapkan **ESP32**, **sensor DHT22**, **breadboard**, dan **kabel jumper**.

### 2. Hubungkan Sensor DHT22

- Sambungkan **pin VCC** dari DHT22 ke **3.3V** ESP32.
- Sambungkan **pin GND** dari DHT22 ke **GND** ESP32.
- Sambungkan **pin data** dari DHT22 ke salah satu **GPIO ESP32**, misalnya **GPIO 17**

### 3. Koneksikan ESP32 ke Komputer

Gunakan **kabel USB** untuk menghubungkan ESP32 ke komputer.

#### 4. Buka Arduino IDE / VS Code

- Pilih **board**: *ESP32 Dev Module*.
- Pilih **port COM** yang sesuai dengan ESP32.
- Pastikan **library DHT** telah diinstal. Jika belum, tambahkan melalui Library Manager dengan mencari "**DHT sensor library**" oleh Adafruit.

#### 5. Tulis dan Unggah Program

```
#include <Arduino.h>

#include <WiFi.h>

#include <HTTPClient.h>

#include "DHT.h"

#define DHTPIN 17

#define DHTTYPE DHT22

DHT dht(DHTPIN, DHTTYPE);

// Ganti dengan kredensial WiFi Anda
const char *ssid = "naruto";
const char *password = "27044869";

unsigned long previousMillis = 0;
const long interval = 5000; // Interval 5 detik (5000 ms)

void setup()
{
  Serial.begin(115200);

  // Hubungkan ke WiFi
  WiFi.begin(ssid, password);

  Serial.print("Menghubungkan ke WiFi");
```

```

while (WiFi.status() != WL_CONNECTED)
{
    delay(500);
    Serial.print(".");
}
Serial.println(" Terhubung!");

dht.begin();

// Tunggu sebentar agar koneksi stabil
delay(1000);
}

void loop()
{
    unsigned long currentMillis = millis();

    // Lakukan POST setiap interval yang telah ditentukan
    if (currentMillis - previousMillis >= interval)
    {
        previousMillis = currentMillis;

        float h = round(dht.readHumidity());
        // Read temperature as Celsius (the default)
        float t = round(dht.readTemperature());

        // Check if any reads failed and exit early (to try again).
        if (isnan(h) || isnan(t))
        {
            Serial.println(F("Failed to read from DHT sensor!"));
            return;
        }
    }
}

```

```

}

// Compute heat index in Celsius (isFahreheit = false)
float hic = dht.computeHeatIndex(t, h, false);

// Inisialisasi HTTPClient
HTTPClient http;

String url = "http://aa1a-114-5-223-135.ngrok-free.app/api/posts"; // Ganti dengan URL ngrok
yang benar

http.begin(url); // Menggunakan HTTP, bukan HTTPS
http.addHeader("Content-Type", "application/json");

String payload = "{\"nama_sensor\":\"Sensor GD\", \"nilai1\":\"" + String(h) + ", \"nilai2\":\"" +
String(t) + "\"}";

Serial.println(payload); // Untuk melihat apakah payload sudah terbentuk dengan benar

// Kirim POST request
int httpResponseCode = http.POST(payload);

// Tampilkan kode respons HTTP
Serial.print("Kode respons HTTP: ");
Serial.println(httpResponseCode);

// Tampilkan respons dari server jika request berhasil
if (httpResponseCode == 200 || httpResponseCode == 201)
{
    String response = http.getString();
    Serial.println("Respons dari server:");
    Serial.println(response);
}

```



```

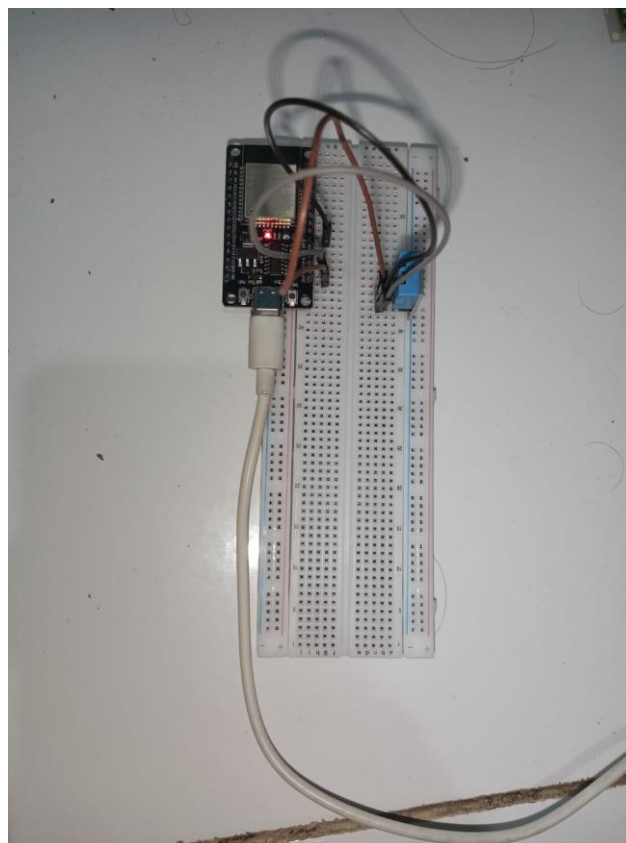
else
{
    Serial.println("Gagal mengirim data");
}

// Tutup koneksi HTTP
http.end();
}
}

```

### 3. Results and Discussion (Hasil dan Pembahasan)

#### 3.1 Experimental Results (Hasil Eksperimen)



<input type="checkbox"/>	 Edit	 Copy	 Delete	24	Sensor GD	4326	359	2025-05-16 06:32:42	2025-05-16 06:32:42
<input type="checkbox"/>	 Edit	 Copy	 Delete	25	Sensor GD	4326	359	2025-05-16 06:32:47	2025-05-16 06:32:47
<input type="checkbox"/>	 Edit	 Copy	 Delete	26	Sensor GD	4326	359	2025-05-16 06:32:52	2025-05-16 06:32:52

Kode berjalan dengan baik dimana datanya terdapat di database

### 4. Appendix (Lampiran)

```

♦ Using ngrok for OSS? Request a community license: https://ngrok.com/r/oss

Session Status      online
Account             Amarxz (Plan: Free)
Update              update available (version 3.22.1, Ctrl-U to update)
Version             3.21.0
Region              Asia Pacific (ap)
Latency             61ms
Web Interface       http://127.0.0.1:4040
Forwarding           http://a1a-114-5-223-135.ngrok-free.app -> http://localhost:8000

Connections          ttl      opn      rt1      rt5      p50      p90
14                  0        0.11     0.04     0.34     0.38

HTTP Requests
-----
13:34:24.989 +07 POST /api/posts      201 Created
13:34:20.103 +07 POST /api/posts      201 Created
13:34:10.050 +07 POST /api/posts      201 Created
13:34:05.152 +07 POST /api/posts      201 Created
13:33:59.996 +07 POST /api/posts      201 Created
13:33:55.099 +07 POST /api/posts      201 Created
13:33:50.030 +07 POST /api/posts      201 Created
13:33:45.279 +07 POST /api/posts      201 Created
13:33:02.479 +07 POST /api/posts      201 Created
13:32:57.381 +07 POST /api/posts      201 Created

```

```

{"nama_sensor": "Sensor GD", "nilai1": 4275.00, "nilai2": 371.00}
Kode respons HTTP: 201
Respons dari server:
{"data":{"id":29,"nama_sensor":"Sensor GD","nilai1":4275,"nilai2":371}}
{"nama_sensor":"Sensor GD", "nilai1":4275.00, "nilai2":359.00}
Kode respons HTTP: 201
Respons dari server:
{"data":{"id":30,"nama_sensor":"Sensor GD","nilai1":4275,"nilai2":359}}
{"nama_sensor":"Sensor GD", "nilai1":4275.00, "nilai2":359.00}
Kode respons HTTP: 201
Respons dari server:
{"data":{"id":31,"nama_sensor":"Sensor GD","nilai1":4275,"nilai2":359}}
{"nama_sensor":"Sensor GD", "nilai1":4288.00, "nilai2":359.00}
Kode respons HTTP: 201
Respons dari server:
{"data":{"id":32,"nama_sensor":"Sensor GD","nilai1":4288,"nilai2":359}}

```

## 2.2 Langkah Implementasi

### 1. Persiapkan Peralatan

Siapkan ESP32, breadboard, dan kabel jumper.

### 2. Hubungkan ESP32 ke Komputer

Gunakan kabel USB untuk menghubungkan ESP32 ke komputer.

### 3. Buka Arduino IDE / VS Code

- Pilih board: **ESP32 Dev Module**.
- Pilih port COM yang sesuai dengan ESP32.

- Pastikan library **WiFi.h** sudah tersedia (ini merupakan library bawaan ESP32).

#### 4. Tulis dan Unggah Program

Gunakan kode berikut untuk memindai jaringan WiFi di sekitar:

```
#include <WiFi.h>

void setup() {
    Serial.begin(115200);

    WiFi.mode(WIFI_STA);
    WiFi.disconnect();
    delay(100);

    Serial.println("Pemindaian Jaringan Wi-Fi Dimulai...");
}

void loop() {
    int n = WiFi.scanNetworks();
    Serial.println("Pemindaian Selesai");
    if (n == 0) {
        Serial.println("Tidak ada jaringan Wi-Fi yang ditemukan.");
    } else {
        Serial.print(n);
        Serial.println(" jaringan Wi-Fi ditemukan:");
        for (int i = 0; i < n; ++i) {
            Serial.print(i + 1);
            Serial.print(": ");
            Serial.print(WiFi.SSID(i));
            Serial.print(" ");
            Serial.print(WiFi.RSSI(i));
            Serial.print("dBm");
            Serial.println((WiFi.encryptionType(i) == WIFI_AUTH_OPEN) ? " " : "*");
        }
    }
}
```

```

        delay(10);
    }
}

Serial.println("");

delay(5000); // Lakukan pemindaian setiap 5 detik
}

```

### 3. Results and Discussion (Hasil dan Pembahasan)

#### 3.1 Experimental Results (Hasil Eksperimen)

##### ▼ TERMINAL

```

2: WiFi-UB.x (-66dBm)*
3: eduroam (-66dBm)*
4: WiFi-UB.x (-74dBm)*
5: eduroam (-74dBm)*
6: WiFi-UB.x (-81dBm)*

```

```

Pemindaian Selesai
8 jaringan Wi-Fi ditemukan:
1: Lab IT (-57dBm)*
2: WiFi-UB.x (-65dBm)*
3: eduroam (-65dBm)*
4: WiFi-UB.x (-73dBm)*
5: eduroam (-74dBm)*
6: eduroam (-82dBm)*
7: WiFi-UB.x (-83dBm)*
8: WiFi-UB.x (-85dBm)*

```

Wifi di sekitar dapat terbaca