LAMPIRAN

1. **Coding**

```
#define BLYNK_PRINT Serial

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

#define TRIG 2 //Module pins
#define ECHO 4

static float kekeruhan;
static float teg;
const int pinS=34;
float pinV;
float tinggi;
const int pompa=25;
const int selenoid=26;

char auth[] = "mtfPyDc1i5wdZHo7h1NyAFBQOk-GEjc3";
char ssid[] = "Klik disini";
char pass[] = "masukansandi";

WidgetLCD lcd(V2);

void setup() {

  Serial.begin(9600);
  pinMode(TRIG, OUTPUT);
  pinMode(ECHO, INPUT_PULLUP);
  Blynk.begin(auth, ssid, pass);
  }

  void loop() {
    Blynk.run();

  //Coding Level Air
```

```
  digitalWrite(TRIG, LOW);
  delayMicroseconds(2);

  digitalWrite(TRIG, HIGH);
  delayMicroseconds(20);

  digitalWrite(TRIG, LOW);
  int jarak = pulseIn(ECHO, HIGH,26000);

  jarak= jarak/58;
  tinggi=70-jarak;

  Serial.print("Tinggi Air ");
  Serial.print(tinggi);
  Serial.println("cm");
  Blynk.virtualWrite(V0, tinggi);
  Blynk.virtualWrite(V2, tinggi);
  lcd.print(0,0,"LevelAir=");
  lcd.print(9,0,tinggi);
  delay(500);

//Coding Kekeruhan Air
  pinV=analogRead(pinS);
  Serial.print("nilai ADC = ");
  Serial.println(pinV);
  delay(500);

  teg=pinV*(5.0/4096);
  Serial.print("Tegangan = ");
  Serial.println(teg);

  kekeruhan = 100.00-(teg/5.1)*100.00;
  Serial.print("nilai kekeruhan = ");
  Serial.print(kekeruhan);
  Blynk.virtualWrite(V1, kekeruhan);
  Blynk.virtualWrite(V3, teg);
  Serial.println(" NTU ");
  delay(500);
```

```
  if(kekeruhan>7){
   if(tinggi>=45){
    digitalWrite(selenoid, LOW);
    digitalWrite(pompa, HIGH);}
   else if(tinggi<=10){
    digitalWrite(selenoid, HIGH);
    digitalWrite(pompa, LOW);}

    Blynk.notify("Tangki dalam Tahap Pengurasan");
    lcd.print(0,1,"ProsesKurasTanki");
   }

   else{
    if(tinggi>=55){
     digitalWrite(selenoid, HIGH);
     digitalWrite(pompa, HIGH);}
    else if(tinggi<=10){
     digitalWrite(selenoid, HIGH);
     digitalWrite(pompa, LOW);}

    lcd.print(0,1,"KondisiAirJernih");
    }
 delay(500);
}
```

2. **Datasheet**
   Datasheet ESP32

1 Overview

ESP32 is a single 2.4 GHz Wi-Fi-and-Bluetooth combo chip designed with the TSMC ultra-low-power 40 nm technology. It is designed to achieve the best power and RF performance, showing robustness, versatility and reliability in a wide variety of applications and power scenarios.

The ESP32 series of chips includes ESP32-D0WD-V3, ESP32-D0WDQ6-V3, ESP32-D0WD, ESP32-D0WDQ6, ESP32-D2WD, ESP32-S0WD, and ESP32-U4WDH, among which, ESP32-D0WD-V3, ESP32-D0WDQ6-V3, and ESP32-U4WDH are based on ECO V3 wafer.

For details on part numbers and ordering information, please refer to Section 7.

For details on ECO V3 instructions, please refer to ESP32 ECO V3 User Guide.

1.1 Featured Solutions

1.1.1 Ultra-Low-Power Solution

ESP32 is designed for mobile, wearable electronics, and Internet-of-Things (IoT) applications. It features all the state-of-the-art characteristics of low-power chips, including fine-grained clock gating, multiple power modes, and dynamic power scaling. For instance, in a low-power IoT sensor hub application scenario, ESP32 is woken up periodically and only when a specified condition is detected. Low-duty cycle is used to minimize the amount of energy that the chip expends. The output of the power amplifier is also adjustable, thus contributing to an optimal trade-off between communication range, data rate and power consumption.

> **Note:**
> For more information, refer to Section 3.7 RTC and Low-Power Management.

1.1.2 Complete Integration Solution

ESP32 is a highly-integrated solution for Wi-Fi-and-Bluetooth IoT applications, with around 20 external components. ESP32 integrates an antenna switch, RF balun, power amplifier, low-noise receive amplifier, filters, and power management modules. As such, the entire solution occupies minimal Printed Circuit Board (PCB) area.

ESP32 uses CMOS for single-chip fully-integrated radio and baseband, while also integrating advanced calibration circuitries that allow the solution to remove external circuit imperfections or adjust to changes in external conditions. As such, the mass production of ESP32 solutions does not require expensive and specialized Wi-Fi testing equipment.

1.2 Wi-Fi Key Features

- 802.11 b/g/n
- 802.11 n (2.4 GHz), up to 150 Mbps
- WMM
- TX/RX A-MPDU, RX A-MSDU
- Immediate Block ACK

- Defragmentation
- Automatic Beacon monitoring (hardware TSF)
- 4 × virtual Wi-Fi interfaces
- Simultaneous support for Infrastructure Station, SoftAP, and Promiscuous modes
  Note that when ESP32 is in Station mode, performing a scan, the SoftAP channel will be changed.
- Antenna diversity

Note:
For more information, please refer to Section 3.5 Wi-Fi.

1.3 BT Key Features

- Compliant with Bluetooth v4.2 BR/EDR and BLE specifications
- Class-1, class-2 and class-3 transmitter without external power amplifier
- Enhanced Power Control
- +12 dBm transmitting power
- NZIF receiver with –94 dBm BLE sensitivity
- Adaptive Frequency Hopping (AFH)
- Standard HCI based on SDIO/SPI/UART
- High-speed UART HCI, up to 4 Mbps
- Bluetooth 4.2 BR/EDR BLE dual mode controller
- Synchronous Connection-Oriented/Extended (SCO/eSCO)
- CVSD and SBC for audio codec
- Bluetooth Piconet and Scatternet
- Multi-connections in Classic BT and BLE
- Simultaneous advertising and scanning

1.4 MCU and Advanced Features

1.4.1 CPU and Memory

- Xtensa® single-/dual-core 32-bit LX6 microprocessor(s), up to 600 MIPS (200 MIPS for ESP32-S0WD/ESP32-U4WDH, 400 MIPS for ESP32-D2WD)
- 448 KB ROM
- 520 KB SRAM
- 16 KB SRAM in RTC
- QSPI supports multiple flash/SRAM chips

1.4.2 Clocks and Timers

- Internal 8 MHz oscillator with calibration
- Internal RC oscillator with calibration
- External 2 MHz ~ 60 MHz crystal oscillator (40 MHz only for Wi-Fi/BT functionality)
- External 32 kHz crystal oscillator for RTC with calibration
- Two timer groups, including 2 × 64-bit timers and 1 × main watchdog in each group
- One RTC timer
- RTC watchdog

1.4.3 Advanced Peripheral Interfaces

- 34 × programmable GPIOs
- 12-bit SAR ADC up to 18 channels
- 2 × 8-bit DAC
- 10 × touch sensors
- 4 × SPI
- 2 × I²S
- 2 × I²C
- 3 × UART
- 1 host (SD/eMMC/SDIO)
- 1 slave (SDIO/SPI)
- Ethernet MAC interface with dedicated DMA and IEEE 1588 support
- Two-Wire Automotive Interface (TWAI®, compatible with ISO11898-1)
- IR (TX/RX)
- Motor PWM
- LED PWM up to 16 channels
- Hall sensor

1.4.4 Security

- Secure boot
- Flash encryption
- 1024-bit OTP, up to 768-bit for customers
- Cryptographic hardware acceleration:
  - AES
  - Hash (SHA-2)

- RSA
- ECC
- Random Number Generator (RNG)

1.5 Applications (A Non-exhaustive List)

- Generic Low-power IoT Sensor Hub
- Generic Low-power IoT Data Loggers
- Cameras for Video Streaming
- Over-the-top (OTT) Devices
- Speech Recognition
- Image Recognition
- Mesh Network
- Home Automation
  - Light control
  - Smart plugs
  - Smart door locks
- Smart Building
  - Smart lighting
  - Energy monitoring
- Industrial Automation
  - Industrial wireless control
  - Industrial robotics
- Smart Agriculture
  - Smart greenhouses
  - Smart irrigation
  - Agriculture robotics
- Audio Applications
  - Internet music players
  - Live streaming devices
  - Internet radio players
  - Audio headsets
- Health Care Applications
  - Health monitoring
  - Baby monitors
- Wi-Fi-enabled Toys
  - Remote control toys
  - Proximity sensing toys
  - Educational toys
- Wearable Electronics
  - Smart watches
  - Smart bracelets
- Retail & Catering Applications
  - POS machines
  - Service robots

1.6 Block Diagram

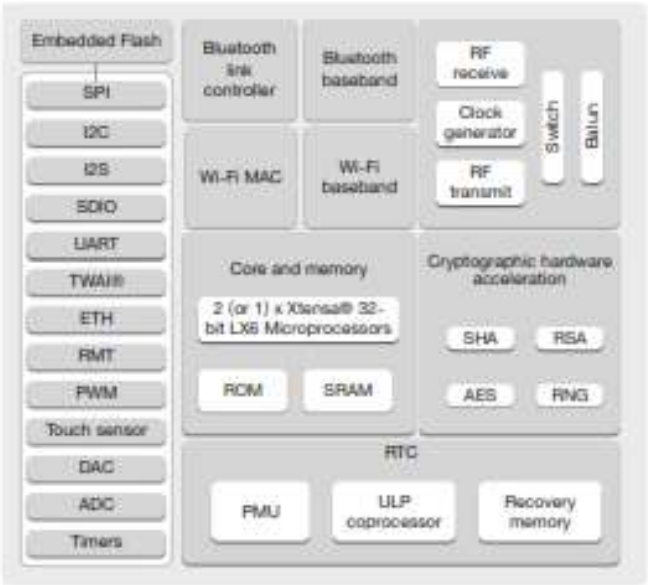

Figure 1: Functional Block Diagram

Note:

Products in the ESP32 series differ from each other in terms of their support for embedded flash and the number of CPUs they have. For details, please refer to Section 7 Part Number and Ordering Information.

2 Pin Definitions

2.1 Pin Layout

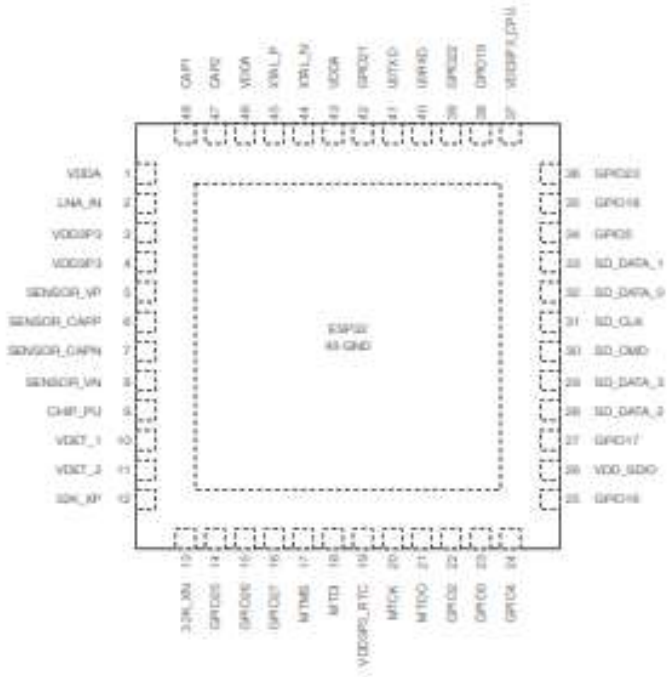

Figure 2: ESP32 Pin Layout (QFN 6*6, Top View)