LAMPIRAN-LAMPIRAN

Lampiran 1 Data Sheet NodeMCU ESP32

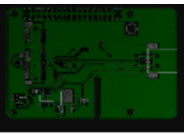
JOY-IT

NodeMCU ESP32

Microcontroller Development Board

Technical Specifications

Model	NodeMCU ESP32
Article No.	SBC-NodeMCU-ESP32
Type	ESP32
Processor	Tensilica LX6 Dual-Core
Clock Frequency	240 MHz
SRAM	512 kB
Memory	4 MB
Wireless Standard	802.11 b/g/n
Frequency	2.4 GHz
Bluetooth	Classic / LE
Data Interfaces	UART / I2C / SPI / DAC / ADC
Operating Voltage	3,3V (operable via 5V-microUSB)
Operating Temperature	-40°C - 125°C
Dimensions (W x D x H)	48 x 26 x 11.5 mm
Scope Of Delivery	NodeMCU ESP32
EAN	4250236816104

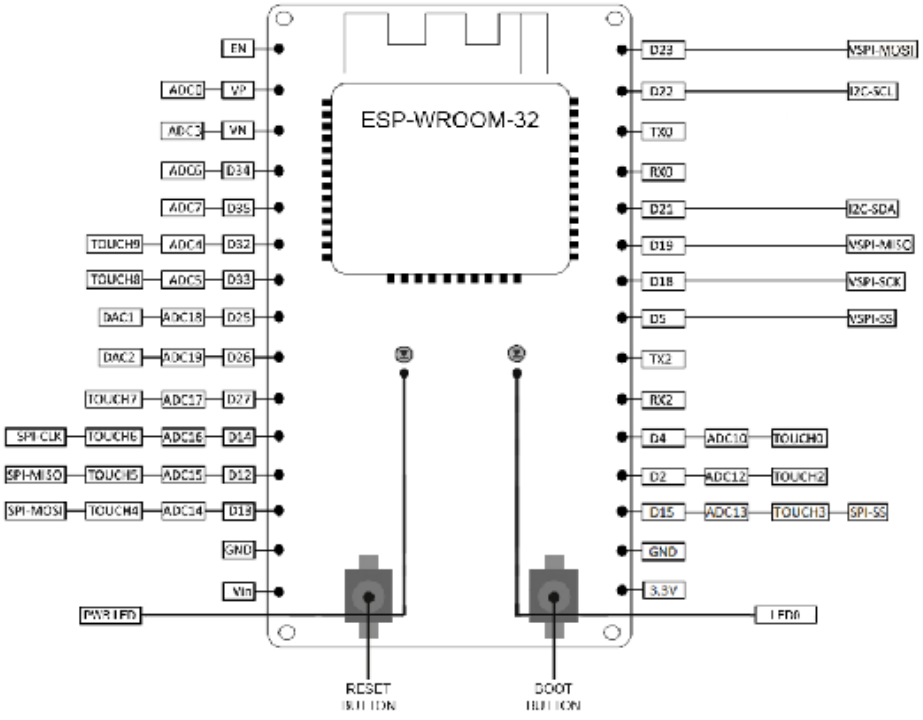


NodeMCU ESP32

Microcontroller Development Board

An overview of the available pins can be seen in the following figure:

2x DAC	15x ADC	1x SPI
1x I²C	2x UART	



Lampiran 2 Data Sheet Water Flow Sensor YF-S201

MODEL: YF-S201

Description:

Water flow sensor consists of a plastic valve body, a water rotor, and a hall-effect sensor. When water flows through the rotor, rotor rolls. Its speed changes with different rate of flow. The hall-effect sensor outputs the corresponding pulse signal. This one is suitable to detect flow in water dispenser or coffee machine. We have a comprehensive line of water flow sensors in different diameters. Check them out to find the one that meets your need most.

Features:

Compact, Easy to Install
High Sealing Performance
High Quality Hall Effect Sensor
RoHS Compliant

Specifications:

Working Voltage: DC 4.5V~24V
Normal Voltage: DC 5V~18V
Max. Working Current: 15mA (DC 5V)
Load capacity: ≤ 10 mA (DC 5V)
Flow Rate Range: 1~30L/min
Load Capacity: ≤10mA (DC 5V)
Operating Temperature: ≤80℃
Liquid Temperature: ≤120℃
Operating Humidity: 35%~90%RH
Allowing Pressure: ≤1.75MPa
Storage Temperature: -25~+ 80℃
Storage Humidity: 25%~95%RH
Electric strength 1250V/min
Insulation resistance ≥ 100MΩ
External threads: 1/2"
Outer diameter: 20mm
Intake diameter: 9mm
Outlet diameter: 12mm

Application:

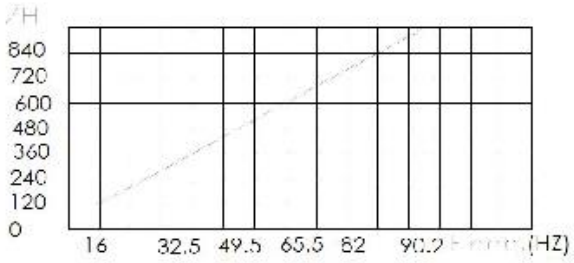
Water heaters, credit card machines, water vending machine, flow measurement device!

Cercuit:

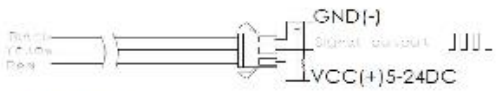
Red: Positive
Black: GND
Yellow: Output signal

Flow Range: 100L/H-/1800H-L/H

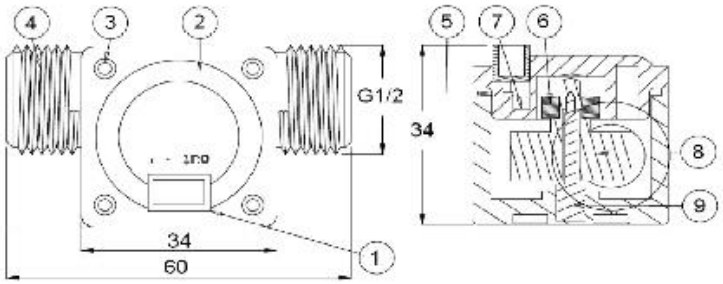
Flow (L/H)	Frezq.(Hz)	Erro range
120	16	±10 5%
240	32.5	
360	49.3	
480	65.5	
600	82	
720	90.2	



Connection method:



N°	Item	Material
1	Wire	PVC
2	Bonnet	PA
3	Screw	Zinc Plated
4	Valve Body	PA
5	Press Valve	
6	Magnet	
7	Hall	
8	Impeller	POM
9	Steel Shaft	SUS304



Closed

Lampiran 3 Data Sheet *Push Button*

Tactile Switches

SPNO

Features:

- Sharp click feel with a positive tactile feedback. Due to small movement distance (stroke), user experiences distinct sensation when the switch "clicks" into place
- Ultra miniature and light weight structure suitable for high density mounting. Economic with high reliability
- Insert moulding in the contact has a special treatment which prevents flux build-up during soldering and permits auto-dipping
- Comprehensive range of SPNO tact switches
- Select by overall height, body, actuator length and operating force

Materials :

Cover	: Stainless steel
Base	: Nylon thermoplastic Colour : Black
Stem	: Nylon thermoplastic Colour : Black (100 gf), Brown (160 gf), Red (260 gf)
Contact disc	: Phosphor bronze with silver cladding
Terminal	: Brass with silver cladding
Cap	: Thermoplastic ABS

Specifications :

Mechanical

Operation Force	: 160 ±50 gf Brown (N) 260 ±50 gf Red (R) 100 ±50 gf Black (K) 350 g Salone (S)
Stop Strength	: Place the switch such that a vertical, static load of 3 kgf shall be applied in the direction of the stem operation for a period of 15 seconds
Stroke	: (12 x 12) 0.35 ±0.1 mm (6 x 6) 0.25 +0.2 / -0.1 mm
Operating Temperature	: -25°C to +70°C
Storage Temperature	: -30°C to +80°C
Vibration Test	: MIL-STD-202F method 201 A Frequency : 10 - 55 - 10 Hz / 1 minute Directions : X, Y, Z three mutually perpendicular directions Time : 2 hours each direction High reliability
Shock Test	: MIL-STD-202F method 213 B Condition A Gravity : 50 G (peak value), 11 msec Direction and Times : 6 sides and 3 times in each direction High reliability

Electrical

Electrical Life	: 260 gf has 50,000 cycles 100 gf, 160 gf has 500,000 cycles

Tactile Switches

SPNO

Specifications :

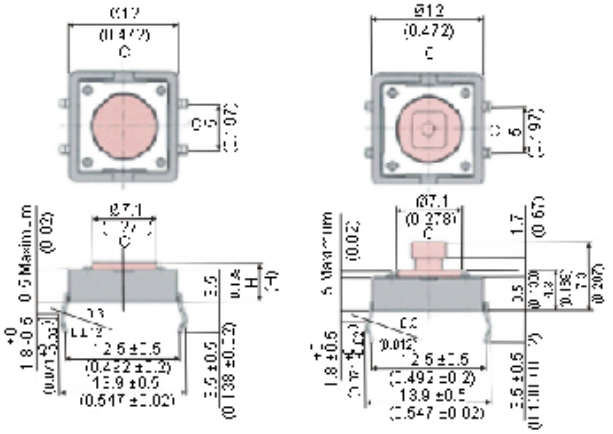
Electrical

Current Rating	: 50 mA, 12 V dc
Maximum Contact Resistance	: 100 mΩ
Insulation Resistance	: 10 MΩ 500 V dc
Dielectric Strength	: 250 V ac / 1 minute
Maximum Wave Solder	: 260°C 5 s 1.6 mm PCB
Hand Solder	: 320°C 2 s 30 W iron
PCB Hole Diameter	: 1 ±0.05 mm

Soldering Process

Wave Soldering : Recommended solder temperature at 500°F (260°C) maximum 5 seconds subject to PCB thickness of 1.6 mm
Hand Soldering : Use a soldering iron of 30 watts, controlled at 608°F (320°C) approximately for 2 seconds while applying solder

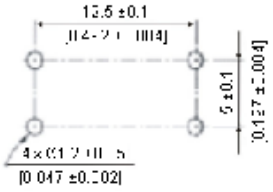
MCDTS-2



Dimensions : Millimetres (Unless Specified)

Circuit

P.C.B. Layout



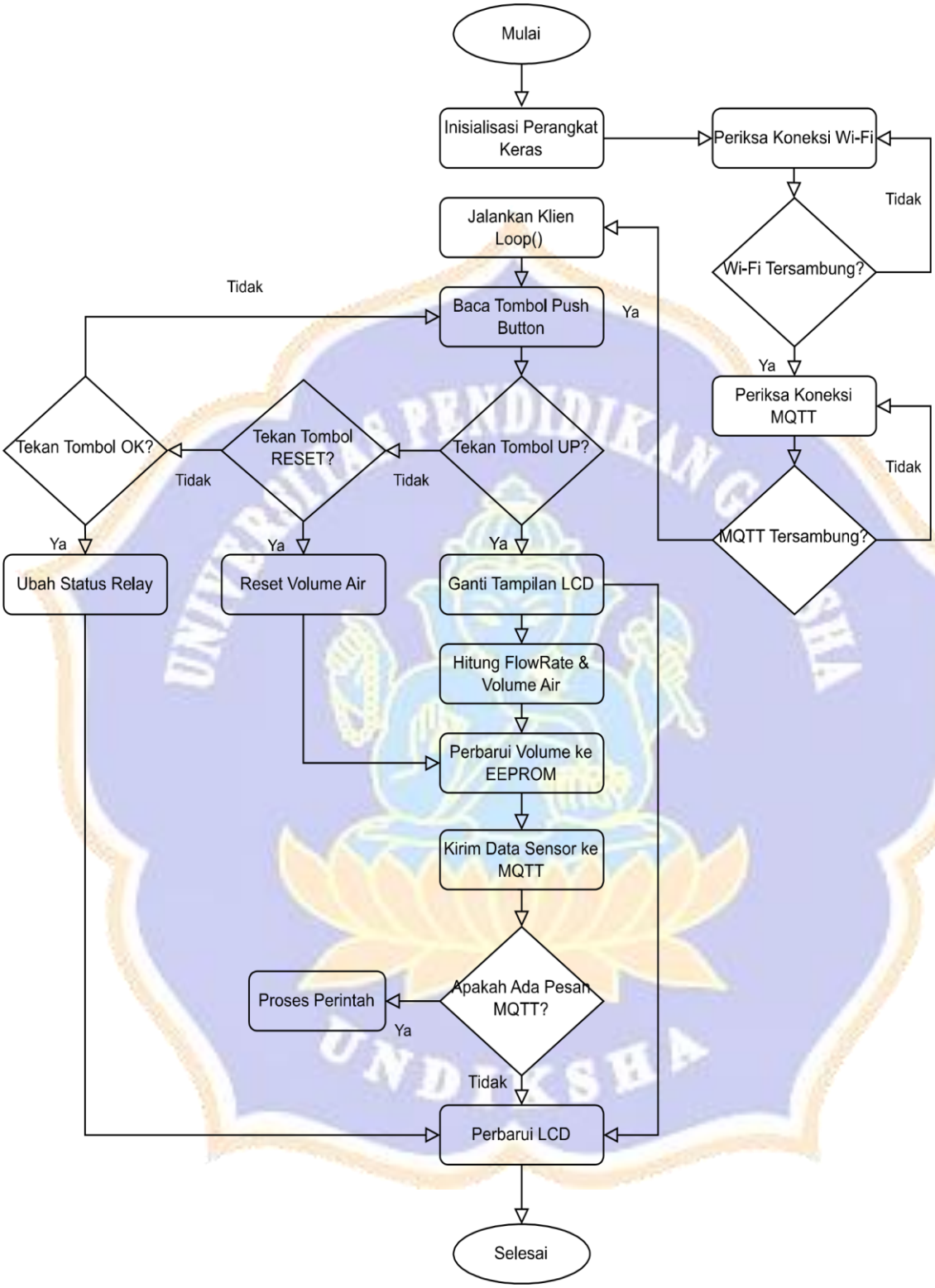
Dimensions : Millimetres (Inches)

Specification Table (12 mm × 12 mm)

Actuator Style / Operating Force (gf)	PCB Drilling	Part Number
7.3 (SQ) × 3.8 / 160	12.5 × 5	MCDTS2-4N

Dimensions : Millimetres (Unless Specified)

Flowchart Program Alat Tugas Akhir



Program Alat Tuas Akhir pada Arduino Ide


```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include <EEPROM.h>
#include <WiFi.h>
#include <PubSubClient.h>
// Pin definitions
#define SENSOR_PIN_1 4
#define SENSOR_PIN_2 16
#define SENSOR_PIN_3 17
#define RELAY_PIN_1 13
#define RELAY_PIN_2 12
#define RELAY_PIN_3 14
#define BUTTON_UP_PIN 27
#define BUTTON_RESET_PIN 26
#define BUTTON_OK_PIN 25
const char* ssid = "OPPO A9 2020";
const char* password = "hendriana03";
const char* mqtt_server = "broker.mqtt-dashboard.com";
WiFiClient espClient;
PubSubClient client(espClient);
// LCD setup
LiquidCrystal_I2C lcd(0x27, 20, 4);
// Constants
const int numSensors = 3;
const int eepromSize = numSensors * 4; // 4 bytes for each totalizer value
const unsigned long updateInterval = 1000; // 1 second
const unsigned long debounceDelay = 200; // 200 milliseconds debounce time for butto
const unsigned long resetDelay = 3000; // 3 seconds for reset button
// Variables
volatile unsigned long pulseCount[numSensors] = {0};
float flowRate[numSensors] = {0.0};
float totalizer[numSensors] = {0};
unsigned long lastUpdateTime = 0;
int currentSensorIndex = 0;
const float calibrationFactor = 5.8;
unsigned long lastDebounceTime = 0;
bool lastButtonUpState = HIGH;
bool lastButtonResetState = HIGH;
bool lastButtonOkState = HIGH;
unsigned long resetButtonPressedTime = 0;
bool isResetting = false;
bool relayState[numSensors] = {false, false, false};
// Define reconnect intervals and timers
unsigned long lastReconnectAttempt = 0;
const unsigned long reconnectInterval = 10000; // Try to reconnect every 10 seconds
// Define unique topics for each sensor
String subscribe = "hendriana/relays";
// Interrupt service routines for flow sensors
void IRAM_ATTR pulseCounter1() { pulseCount[0]++; }
void IRAM_ATTR pulseCounter2() { pulseCount[1]++; }
void IRAM_ATTR pulseCounter3() { pulseCount[2]++; }
```

```
void setup() {
  // Initialize serial communication
  Serial.begin(115200);
  // Initialize LCD
  lcd.begin(20, 4);
  lcd.init();
  lcd.backlight();
  // Initialize buttons
  pinMode(BUTTON_UP_PIN, INPUT_PULLUP);
  pinMode(BUTTON_RESET_PIN, INPUT_PULLUP);
  pinMode(BUTTON_OK_PIN, INPUT_PULLUP);
  // Initialize flow sensor pins and attach interrupts
  pinMode(SENSOR_PIN_1, INPUT_PULLUP);
  attachInterrupt(digitalPinToInterrupt(SENSOR_PIN_1), pulseCounter1, FALLING);
  pinMode(SENSOR_PIN_2, INPUT_PULLUP);
  attachInterrupt(digitalPinToInterrupt(SENSOR_PIN_2), pulseCounter2, FALLING);
  pinMode(SENSOR_PIN_3, INPUT_PULLUP);
  attachInterrupt(digitalPinToInterrupt(SENSOR_PIN_3), pulseCounter3, FALLING);
  // Initialize relay pins
  pinMode(RELAY_PIN_1, OUTPUT);
  pinMode(RELAY_PIN_2, OUTPUT);
  pinMode(RELAY_PIN_3, OUTPUT);
  digitalWrite(RELAY_PIN_1, HIGH); // Turn off relay
  digitalWrite(RELAY_PIN_2, HIGH); // Turn off relay
  digitalWrite(RELAY_PIN_3, HIGH); // Turn off relay
  // Initialize EEPROM
  EEPROM.begin(eepromSize);
  // Load totalizer values from EEPROM dengan skala dua desimal
  for (int i = 0; i < numSensors; i++) {
    int scaledTotalizer = EEPROM.readUInt(i * 4);
    totalizer[i] = scaledTotalizer / 100.0; // Kembalikan ke float asli dengan dua d
  }
  lcd.setCursor(0, 0);
  lcd.print("MONITORING DEBIT AIR");
  lcd.setCursor(0, 1);
  lcd.print("      D4 TRSE       ");
  lcd.setCursor(0, 2);
  lcd.print("      UNDIKSHA      ");
  lcd.setCursor(0, 3);
  lcd.print("        2024        ");
  delay(3000);
  lcd.clear();
  setup_wifi();
  client.setServer(mqtt_server, 1883);
  client.setCallback(callback);
}
void loop() {
  // Check if Wi-Fi is connected
  if (WiFi.status() != WL_CONNECTED) {
    unsigned long currentMillis = millis();
    // Only try to reconnect after the interval has passed
    if (currentMillis - lastReconnectAttempt > reconnectInterval) {
      lastReconnectAttempt = currentMillis;
      setup_wifi(); // Attempt to reconnect Wi-Fi
    }
  } else {
    // If Wi-Fi is connected, check if MQTT is connected
    if (!client.connected()) {
      unsigned long currentMillis = millis();
      if (currentMillis - lastReconnectAttempt > reconnectInterval) {
        lastReconnectAttempt = currentMillis;
        reconnect(); // Attempt to reconnect MQTT
      }
    } else {
      // MQTT is connected, process MQTT messages
```

```
      client.loop();
    }
  }
  unsigned long currentTime = millis();
  unsigned long elapsedTime = currentTime - lastUpdateTime;
  handleButtons(currentTime);
  updateFlowData(elapsedTime);
  updateLCD();
}
void handleButtons(unsigned long currentTime) {
  bool buttonUpState = digitalRead(BUTTON_UP_PIN);
  bool buttonResetState = digitalRead(BUTTON_RESET_PIN);
  bool buttonOkState = digitalRead(BUTTON_OK_PIN);
  // UP button to change sensor display
  if (buttonUpState != lastButtonUpState && (currentTime - lastDebounceTime) > debou
    if (buttonUpState == LOW) {
      currentSensorIndex = (currentSensorIndex + 1) % numSensors;
      updateLCD();
      lastDebounceTime = currentTime;
    }
  }
  lastButtonUpState = buttonUpState;

  // Reset button to reset totalizer
  if (buttonResetState == LOW && !isResetting) {
    resetButtonPressedTime = currentTime;
    isResetting = true;
  }
  if (buttonResetState == LOW && isResetting && (currentTime - resetButtonPressedTim
    totalizer[currentSensorIndex] = 0;
    EEPROM.writeUInt(currentSensorIndex * 4, 0);
    EEPROM.commit();
    updateLCD();
    isResetting = false;
  }
  if (buttonResetState == HIGH) {
    isResetting = false;
  }
  lastButtonResetState = buttonResetState;
  // Manual control with OK button
  if (buttonOkState != lastButtonOkState && (currentTime - lastDebounceTime) > debou
    if (buttonOkState == LOW) {
      if (currentSensorIndex == 0) {
        relayState[0] = !relayState[0];  // Toggle Relay 1 state
        digitalWrite(RELAY_PIN_1, relayState[0] ? LOW : HIGH); // Set relay pin stat
      } else if (currentSensorIndex == 1) {
        relayState[1] = !relayState[1];  // Toggle Relay 2 state
        digitalWrite(RELAY_PIN_2, relayState[1] ? LOW : HIGH); // Set relay pin stat
      } else if (currentSensorIndex == 2) {
        relayState[2] = !relayState[2];  // Toggle Relay 3 state
        digitalWrite(RELAY_PIN_3, relayState[2] ? LOW : HIGH); // Set relay pin stat
      }
      // Publish the new state to MQTT
      String relayCommand = "relay" + String(currentSensorIndex + 1) + (relayState[c
      client.publish("hendriana/relays", relayCommand.c_str());
      // Update LCD to reflect the change
      updateLCD();
      lastDebounceTime = currentTime;
    }
  }
  lastButtonOkState = buttonOkState;
}
```

```
void updateFlowData(unsigned long elapsedTime) {
  if (elapsedTime >= updateInterval) {
    for (int i = 0; i < numSensors; i++) {
      flowRate[i] = ((1000.0 / elapsedTime) * pulseCount[i]) / calibrationFactor;
      totalizer[i] += (flowRate[i] / 60.0);
      if (totalizer[i] >= 999999) {
        totalizer[i] = 0;
      }
      pulseCount[i] = 0;
      // Skala nilai totalizer untuk menyimpan ke EEPROM dengan dua desimal
      int scaledTotalizer = totalizer[i] * 100; // Skala dua tempat desimal
      EEPROM.writeUInt(i * 4, scaledTotalizer);
    }
    EEPROM.commit();
    updateLCD();
    lastUpdateTime = millis();
    if (WiFi.status() == WL_CONNECTED) {
      sendToMQTT();
    }
  }
}
void sendToMQTT() {
  // Publish flow rate and totalizer to MQTT
  char flowString1[8], flowString2[8], flowString3[8];
  char totalizerString1[8], totalizerString2[8], totalizerString3[8];
    dtostrf(flowRate[0], 1, 2, flowString1);
    dtostrf(flowRate[1], 1, 2, flowString2);
    dtostrf(flowRate[2], 1, 2, flowString3);
    dtostrf(totalizer[0], 1, 2, totalizerString1);
    dtostrf(totalizer[1], 1, 2, totalizerString2);
    dtostrf(totalizer[2], 1, 2, totalizerString3);
    client.publish("hendriana/sensor1/flow", flowString1);
    client.publish("hendriana/sensor2/flow", flowString2);
    client.publish("hendriana/sensor3/flow", flowString3);
    client.publish("hendriana/sensor1/volume", totalizerString1);
    client.publish("hendriana/sensor2/volume", totalizerString2);
    client.publish("hendriana/sensor3/volume", totalizerString3);
}
void setup_wifi() {
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);
  WiFi.mode(WIFI_STA);
  WiFi.begin(ssid, password);
  unsigned long startTime = millis();
  while (WiFi.status() != WL_CONNECTED && millis() - startTime < 1000) {
    delay(500);
    Serial.print(".");
    updateLCD();
  }
  Serial.println(WiFi.status() == WL_CONNECTED ? "\nWiFi connected" : "\nWiFi not co
}
void callback(char* topic, byte* payload, unsigned int length) {
  Serial.print("Message arrived [");
  Serial.print(topic);
  Serial.print("] ");
  String payloadString = "";
  for (int i = 0; i < length; i++) {
    payloadString += (char)payload[i];
  }
  Serial.println(payloadString);
  if (payloadString == "relay1_on") {
    relayState[0] = true;
    digitalWrite(RELAY_PIN_1, LOW);
    Serial.println("Relay 1 ON");
```

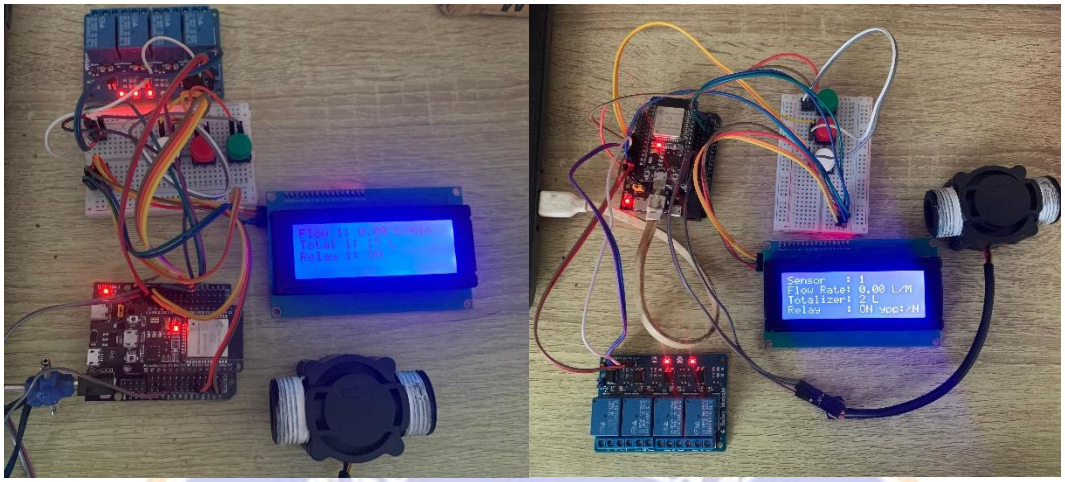
```
  } else if (payloadString == "relay1_off") {
    relayState[0] = false;
    digitalWrite(RELAY_PIN_1, HIGH);
    Serial.println("Relay 1 OFF");
  } else if (payloadString == "relay2_on") {
    relayState[1] = true;
    digitalWrite(RELAY_PIN_2, LOW);
    Serial.println("Relay 2 ON");
  } else if (payloadString == "relay2_off") {
    relayState[1] = false;
    digitalWrite(RELAY_PIN_2, HIGH);
    Serial.println("Relay 2 OFF");
  } else if (payloadString == "relay3_on") {
    relayState[2] = false;
    digitalWrite(RELAY_PIN_3, HIGH);
    Serial.println("Relay 3 OFF");
  }
  // Reset totalizer with 3-second delay for consistency with manual reset
  else if (payloadString == "reset_totalizer1") {
    lcd.setCursor(0, 2);
    lcd.print("Volume  : Reset...");
    delay(3000); // 3-second delay
    totalizer[0] = 0;
    EEPROM.writeUInt(0, 0);
    EEPROM.commit();
    Serial.println("Totalizer 1 Reset via MQTT");
  } else if (payloadString == "reset_totalizer2") {
    lcd.setCursor(0, 2);
    lcd.print("Volume  : Reset...");
    delay(3000); // 3-second delay
    totalizer[1] = 0;
    EEPROM.writeUInt(4, 0);
    EEPROM.commit();
    Serial.println("Totalizer 2 Reset via MQTT");
  } else if (payloadString == "reset_totalizer3") {
    lcd.setCursor(0, 2);
    lcd.print("Volume  : Reset...");
    delay(3000); // 3-second delay
    totalizer[2] = 0;
    EEPROM.writeUInt(8, 0);
    EEPROM.commit();
    Serial.println("Totalizer 3 Reset via MQTT");
  }
  updateLCD();
}
void updateLCD() {
  lcd.setCursor(0, 0);
  lcd.print("Sensor  : ");
  lcd.print(currentSensorIndex + 1);
  lcd.setCursor(0, 1);
  lcd.print("FlowRate: ");
  lcd.print(flowRate[currentSensorIndex]);
  lcd.print(" L/M ");
  lcd.setCursor(0, 2);
  lcd.print("Volume  : ");
  lcd.print(totalizer[currentSensorIndex]);
  lcd.print(" L  ");
  if (isResetting) {
    lcd.setCursor(0, 2);
    lcd.print("Volume  : Reset...");
  } else {
    lcd.setCursor(0, 3);
    lcd.print("Relay   : ");
    lcd.print(relayState[currentSensorIndex] ? "ON " : "OFF");
  }
```

```
}
void reconnect() {
  while (!client.connected()) {
    Serial.print("Attempting MQTT connection...");
    String clientId = "ESP32Client-";
    clientId += String(random(0xffff), HEX);
    if (client.connect(clientId.c_str())) {
      Serial.println("connected");
      client.publish("hendriana/msg", "ONLINE !");
      client.subscribe(subscribe.c_str());
    } else {
      Serial.print("failed, rc=");
      Serial.print(WiFi.status());
      Serial.println(" try again in 5 seconds");
      delay(5000);
    }
  }
}
```

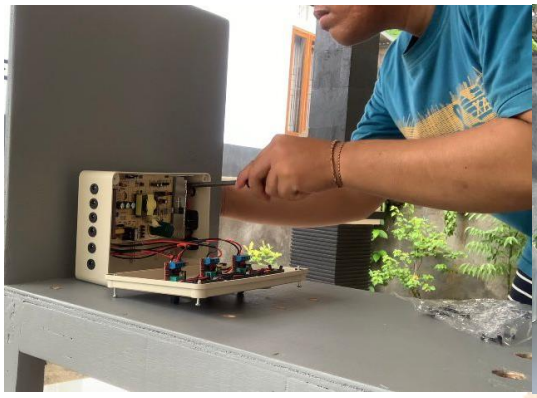
```
Invalid library found in C:\Users\ACER\Documents\Arduino\libraries\thnispeakcode: no
```

Lampiran Dokumentasi Perancangan, Pembuatan, Pengujian Alat, serta Hasil Alat

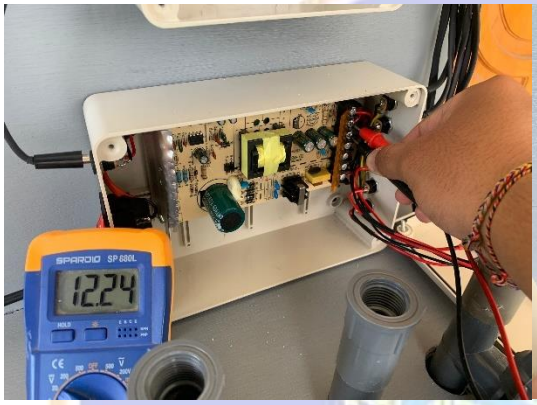
1. Dokumentasi Perancangan Alat

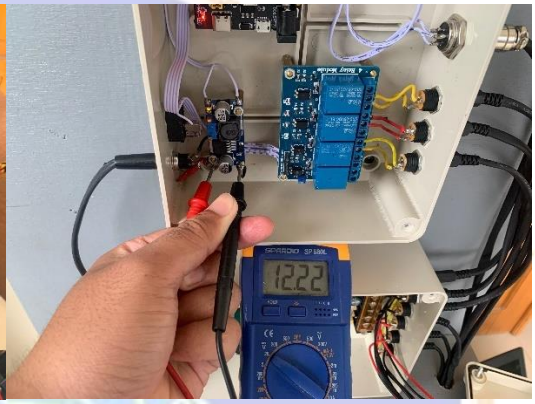


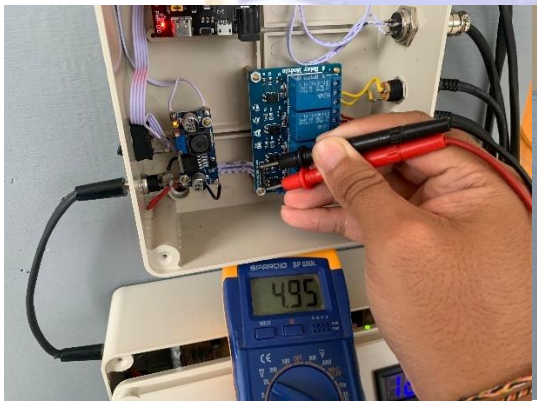
2. Dokumentasi Pembuatan Alat




3. Dokumentasi Pengujian Alat







4. Dokumentasi Hasil Alat

KEMENTERIAN PENDIDIKAN, KEBUDAYAAN,
RISET, DAN TEKNOLOGI
UNIVERSITAS PENDIDIKAN GANESHA
FAKULTAS TEKNIK DAN KEJURUAN
Alamat : Jalan Udayana No. 11 Singaraja
Telepon-Fax: (0362) 22570Kode Pos. 81116
Website : www.undiksha.ac.id

FORMULIR BIMBINGAN TUGAS AKHIR

Nama	:	Putu Hendriana
Nim	:	2255023003
Prodi	:	D4 Teknologi Rekayasa Sistem Elektronika
Pembimbing	:	Dr. Gede Indrawan, S.T., M.T.
Judul	:	Pengembangan Sistem Kendali dan Monitoring Meteran Air Menggunakan Node MCU Berbasis Internet Of Things (IOT) Pada Saluran Air

Pembahasan Mahasiswa :

- Proses Integrasi dari alat ke IOT

Pembahasan Pembimbing :

Cek berbagai platform IoT yg fiturnya masih banyak gratis-nya

Hari/Tgl	TDD Mahasiswa	TDD Pembimbing
Selasa, 29 Oktober 2024	Putu Hendriana	

KEMENTERIAN PENDIDIKAN, KEBUDAYAAN,
RISET, DAN TEKNOLOGI
UNIVERSITAS PENDIDIKAN GANESHA
FAKULTAS TEKNIK DAN KEJURUAN
Alamat : Jalan Udayana No. 11 Singaraja
Telepon-Fax: (0362) 22570Kode Pos. 81116
Website : www.undiksha.ac.id

FORMULIR BIMBINGAN TUGAS AKHIR

Nama	:	Putu Hendriana
Nim	:	2255023003
Prodi	:	D4 Teknologi Rekayasa Sistem Elektronika
Pembimbing	:	Dr. Gede Indrawan, ST. MT
Judul	:	Pengembangan Sistem Kendali dan Monitoring meteran Air menggunakan NodeMCU Berbasis Internet Of Things (IOT) Pada Saluran Air

Pembahasan Mahasiswa :

- Proses Akurasi data aliran air pada sensor flow water

Pembahasan Pembimbing :

Lanjutkan untuk proses akurasi data

Hari/Tgl	TDD Mahasiswa	TDD Pembimbing
Jumat, 8 November 2024		

KEMENTERIAN PENDIDIKAN, KEBUDAYAAN,
RISET, DAN TEKNOLOGI
UNIVERSITAS PENDIDIKAN GANESHA
FAKULTAS TEKNIK DAN KEJURUAN
Alamat : Jalan Udayana No. 11 Singaraja
Telepon-Fax: (0362) 22570Kode Pos. 81116
Website : www.undiksha.ac.id

FORMULIR BIMBINGAN TUGAS AKHIR

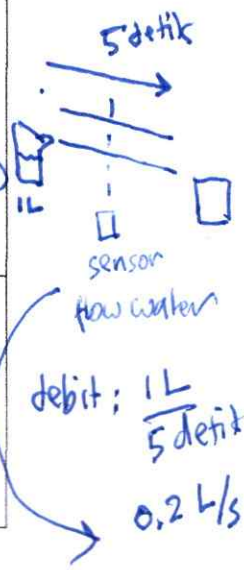
Nama	:	Putu Hendriana
Nim	:	2255023003
Prodi	:	D4 Teknologi Rekayasa Sistem Elektronika
Pembimbing	:	Dr. Gede Indrawan, ST, MT
Judul	:	Pengembangan Sistem, Kendali dan Monitoring Meteran Air Menggunakan Node MCU Berbasis Internet of Things (IOT) Pada Saluran Air

Pembahasan Mahasiswa :

- Proses Pengemasan alat
- Proses Kalibrasi sensor flow water

Pembahasan Pembimbing :

- Casing sudah dibuat, minggu ini akan diintegrasikan dengan alat yg dibuat
- kalibrasi perlu pembanding dengan ground truth (perlu dipikirkan alat pembanding yg memberikan nilai ukur yg benar)



Hari/Tgl	TDD Mahasiswa	TDD Pembimbing
Selasa, 26 November 2024		

KEMENTERIAN PENDIDIKAN, KEBUDAYAAN,
RISET, DAN TEKNOLOGI
UNIVERSITAS PENDIDIKAN GANESHA
FAKULTAS TEKNIK DAN KEJURUAN
Alamat : Jalan Udayana No. 11 Singaraja
Telepon-Fax: (0362) 22570Kode Pos. 81116
Website : www.undiksha.ac.id

FORMULIR BIMBINGAN TUGAS AKHIR

Nama	:	Putu Hendriana
Nim	:	2255023003
Prodi	:	D4 Teknologi Rekayasa Sistem Elektronika
Pembimbing	:	Dr. Gede Indrawan, ST, MT
Judul	:	Pengembangan Sistem Kendali dan Monitoring Meteran Air Menggunakan NodeMCU Berbasis Internet of Things (IOT) Pada Saluran Air.

Pembahasan Mahasiswa :

- Pengujian masing-masing modul dan pengambilan data pada sensor flow water

Pembahasan Pembimbing :

- Sudah ada metode validasi untuk akurasi pengukuran

Hari/Tgl	TDD Mahasiswa	TDD Pembimbing
Selasa, 10 Desember 2024		

RIWAYAT HIDUP

Putu Hendriana lahir di Temukus pada tanggal 09 Januari 2003. Penulis lahir dari pasangan suami istri Bapak I Nyoman Budiana dan Ibu Ni Kadek Intariani. Penulis berkebangsaan Indonesia dan beragama Hindu. Kini penulis beralamat di Banjar Dinas Tengah, Desa Temukus, Kecamatan Banjar, Kabupaten Buleleng, Provinsi Bali. Penulis menyelesaikan pendidikan dasar di SD Negeri 4 Temukus dan lulus pada tahun 2015. Kemudian penulis melanjutkan di SMP Negeri 3 Banjar dan lulus pada tahun 2018. Pada tahun 2021, penulis lulus dari SMK Negeri 3 Singaraja jurusan Teknik Audio Video dan melanjutkan ke Diploma 4 Jurusan Teknologi Industri Program Studi Teknologi Rekayasa Sistem Elektronika di Universitas Pendidikan Ganesha. Pada semester akhir tahun 2024 penulis telah menyelesaikan Tugas Akhir yang berjudul "Pengembangan Sistem Kendali dan *Monitoring* Meteran Air Menggunakan NodeMCU Berbasis *Internet of Things (IoT)* Pada Saluran Air". Selanjutnya, mulai tahun 2024 sampai dengan penulisan skripsi ini, penulis masih terdaftar sebagai mahasiswa Program Diploma 4 Teknologi Rekayasa Sistem Elektronika di Universitas Pendidikan Ganesha.