```
#include <LiquidCrystal_I2C.h>            // Library untuk LCD I2C
#include <SoftwareSerial.h>               // Library untuk komunikasi serial tambahan
#include <PCD8544.h>                      // Library untuk LCD Nokia 5110 (PCD8544)
#include <Wire.h>                         // Library untuk komunikasi I2C

PCD8544 lcd(3,4,5,7,6);                   // Inisialisasi PIN LCD Nokia 5110
SoftwareSerial myArd (19, 18);            // RxArd TxArd  19D1 18D2

String idMeter = "2";                     // ID unik untuk meteran

volatile int flow_frequency;              // Frekuensi pulsa dari sensor aliran
unsigned int l_hour;                      // Liter per jam
unsigned long currentTime;                // Waktu saat ini
unsigned long cloopTime;                  // Waktu loop sebelumnya
float ml_air;                             // Volume air dalam mililiter

volatile int pulseCount = 0;              // Jumlah pulsa yang dihitung
float flowRate = 0.0;                     // Laju aliran dalam L/min
unsigned int flowLitres = 0;              // Volume air dalam liter
unsigned int flowMilliLitres = 0;         // Volume air dalam mililiter
unsigned long totalMilliLitres = 0;       // Total volume air dalam mililiter
unsigned long startTime;                  // Waktu mulai pengukuran

// Pin Data
int flowPin   = 2;                        // Pin untuk sensor aliran
int echoPin   = 10;                       // Pin echo untuk sensor ultrasonik
int trigPin   = 11;                       // Pin trigger untuk sensor ultrasonik
int tegangan  = A0;                       // Pin analog untuk pembacaan tegangan
int pinValve8 = 14;                       // Pin kontrol valve 8
int pinValve9 = 15;                       // Pin kontrol valve 9
int miringPin = 20;                       // Pin untuk sensor kemiringan
int buzzerPin = 17;                       // Pin untuk buzzer

// Default Value
float nilaiPulsaLCD = 0.0;                // Nilai pulsa untuk ditampilkan di LCD
float dataPUL       = 0.0;                // Data pulsa yang diterima
bool buzzerTerusan  = false;              // Status buzzer menyala terus
bool setAwal        = false;              // Status pengaturan awal
bool setSetup       = false;              // Status pengaturan setup
bool setNilai       = false;              // Status pengaturan nilai
bool setWifiku      = false;              // Status pengaturan WiFi
bool nivalve        = false;              // Status valve
bool nivolt         = false;              // Status tegangan
bool nivoltbuka     = false;              // Status tegangan untuk membuka valve
bool cekPulsaValve  = false;              // Status pengecekan pulsa untuk valve
bool cekPulsaValve2 = true;               // Status pengecekan pulsa untuk valve2
bool kirimHabis     = false;              // Status pengiriman data saat pulsa habis
bool infoKedip      = false;              // Status informasi kedipan
bool infoHabis      = false;              // Status informasi pulsa habis
bool cekPintu       = false;              // Status pengecekan pintu
bool cekValveTutup  = false;              // Status pengecekan valve tertutup
String NPku;                              // Nomor pelanggan
int jarakToleransi  = 15;                 // Toleransi jarak untuk sensor ultrasonik
float teganganVolt;                       // Nilai tegangan yang dibaca
```

```ino
float Volt;                                 // Tegangan dalam volt
float pemakaian = 0;                        // Total pemakaian air
long duration;                              // Durasi pulsa dari sensor ultrasonik
int distance;                               // Jarak yang diukur oleh sensor ultrasonik

// Fungsi interrupt untuk menghitung frekuensi aliran air
void flow(){
  flow_frequency++;
}

// Fungsi interrupt untuk menghitung jumlah pulsa dari sensor aliran
void pulseCounter() {
  pulseCount++;
}

// Fungsi setup()
// ----------------------------------------------
// Fungsi ini dijalankan satu kali saat Arduino dinyalakan
// atau setelah tombol reset ditekan.
//
// Tujuan utama fungsi ini adalah untuk:
// - Menginisialisasi pin sebagai input atau output
// - Mengatur nilai awal untuk pin digital
// - Memulai komunikasi serial antara Arduino dan perangkat lain (misal ESP)
// - Inisialisasi modul tambahan seperti LCD, sensor, dan valve
//
// Semua pengaturan awal sistem, seperti mode pin, status awal valve,
// dan tampilan LCD dilakukan di sini agar sistem siap beroperasi.
// ----------------------------------------------
void setup() {
  Serial.begin(9600);       // Inisialisasi komunikasi serial utama
  Serial1.begin(9600);      // Inisialisasi komunikasi serial tambahan
  myArd.begin(9600);        // Inisialisasi komunikasi serial dengan Arduino tambahan
  delay(2000);              // Delay untuk stabilisasi

  Serial.println("init");   // Menampilkan pesan inisialisasi
  Serial1.println("init");  // Menampilkan pesan inisialisasi

  pinMode(flowPin, INPUT);          // Mengatur pin sensor aliran sebagai input
  digitalWrite(flowPin, HIGH);      // Menyalakan pull-up resistor
  // attachInterrupt(0, flow, RISING);
  attachInterrupt(digitalPinToInterrupt(flowPin), pulseCounter, FALLING); // Mengatur
interrupt pada pin sensor aliran
  startTime = millis();             // Menyimpan waktu mulai

  sei();                            // Mengaktifkan interrupt global
  currentTime = millis();           // Menyimpan waktu saat ini
  cloopTime = currentTime;          // Menyimpan waktu loop sebelumnya

  pinMode(trigPin, OUTPUT);         // Mengatur pin trigger sebagai output
  pinMode(echoPin, INPUT);          // Mengatur pin echo sebagai input
  pinMode(pinValve8, OUTPUT);       // Mengatur pin valve 8 sebagai output
  pinMode(pinValve9, OUTPUT);       // Mengatur pin valve 9 sebagai output
  pinMode(miringPin, INPUT);        // Mengatur pin sensor kemiringan sebagai input
```

```
  pinMode(buzzerPin, OUTPUT);      // Mengatur pin buzzer sebagai output

  lcd.begin(84, 48);               // Inisialisasi LCD Nokia 5110
  lcd.setCursor(0, 0);             // Mengatur posisi kursor
  lcd.print("   WELCOME");         // Menampilkan pesan WELCOME
  lcd.setCursor(0, 2);             // Mengatur posisi kursor
  lcd.print("  LING SMART");       // Menampilkan pesan LING SMART
  lcd.setCursor(0, 3);             // Mengatur posisi kursor
  lcd.print("   SOLUTION");        // Menampilkan pesan SOLUTION
  lcd.setCursor(0, 5);             // Mengatur posisi kursor
  lcd.print("-----SSA------");     // Menampilkan pesan -----SSA-----
  delay(3000);                     // Delay untuk menampilkan pesan
}
```

```
// Fungsi loop()
// ------------------------------------------------
// Fungsi ini akan dijalankan **berulang kali tanpa henti** selama Arduino menyala.
//
// Di sinilah letak **logika utama program**, seperti:
// - Membaca data sensor (aliran air, pintu, kemiringan, tegangan, dll.)
// - Mengontrol aktuator (valve, buzzer)
// - Menampilkan informasi ke LCD
// - Mengirim atau menerima data melalui komunikasi serial
// - Melakukan perhitungan pulsa air, konversi debit, dan pengurangan saldo
//
// Fungsi ini akan terus memantau sistem secara real-time dan
// mengambil tindakan jika ditemukan kondisi yang memerlukan respon.
//
// Semua pemrosesan sensor dan kontrol terjadi di dalam loop()
// agar sistem bisa berjalan terus tanpa henti.
// ------------------------------------------------
void loop() {
  jarakTutup();      // Memanggil fungsi jarakTutup() untuk kontrol valve
  cekTegangan();     // Memanggil fungsi cekTegangan() untuk control valve


  String msg = Serial1.readStringUntil('\n'); // Membaca pesan dari Serial1 hingga newline
  msg.trim();               // Menghapus spasi di awal dan akhir
  Serial.println(msg);      // Menampilkan pesan yang diterima

  // Jika pesan yang diterima adalah "GetData"
  // Jalankan fungsi di dalam if
  if (msg.equals("GetData")) {
    Serial.println(setNilai);      // Tampilkan data dari variable setNilai
    if (setNilai == false) {
      lcd.clear();                 // Hapus atau bersihkan LCD
      lcd.setCursor(3, 2);         // Mengatur kursor pada titik tertentu
      lcd.print("CONNECTING....."); // Menampilkan tulisan CONNECTING...
      delay(3000);                 // Delay dengan durasi tertentu
      setNilai = true;             // Ubah nilai dari setNilai menjadi true
    }
      Serial.println("Masuk Sini");
      String dataKU = Serial1.readString();      // Membaca data dari Serial1
      dataKU.trim();                             // Hapus spasi dari dataKU
      Serial.println(dataKU);

      // Parsing data pelanggan dan pulsa dari dataKU
      // Ambil posisi awal dan akhir dari Nomor Pelanggan dalam string
                    // Posisi mulai tag "#@" (sebelum nomor pelanggan)
      int indexSNP = dataKU.indexOf("#@");
                    // Posisi akhir tag "#&" (sebelum nama pelanggan)
      int indexLNP = dataKU.lastIndexOf("#&");

      // Ekstrak substring dari setelah "#@" sampai sebelum "#&"
      String dataNP = dataKU.substring(indexSNP+2, indexLNP);
      NPku = dataNP; // Simpan hasil ekstraksi ke variabel NPku
```

```
// Ambil posisi awal dan akhir dari Nama Pelanggan dalam string
                // Posisi mulai tag "#&" (sebelum nama pelanggan)
int indexSNM = dataKU.indexOf("#&");
                // Posisi akhir tag "#!" (sebelum nilai pulsa)
int indexLNM = dataKU.lastIndexOf("#!");

// Ekstrak substring dari setelah "#&" sampai sebelum "#!"
String dataNM = dataKU.substring(indexSNM+2, indexLNM);

// Ambil posisi awal dan akhir dari Nilai Pulsa dalam string
                // Posisi mulai tag "#!" (sebelum nilai pulsa)
int indexSPL = dataKU.indexOf("#!");
                // Posisi akhir tag "##" (akhir data)
int indexLPL = dataKU.lastIndexOf("##");



// Ekstrak substring dari setelah "#!" sampai sebelum "##"
String dataPL = dataKU.substring(indexSPL+2, indexLPL);

dataPL.trim();        // Bersihkan spasi atau newline dari nilai pulsa
dataPL.toFloat();     // Konversi string nilai pulsa ke float

// Hitung nilai pulsa dalam liter air: misalnya 3400 = Rp 1000
double convertPulsa = (dataPL.toInt() / 3400.0) * 1000;

dataPUL = convertPulsa;      // Simpan hasil konversi ke variabel global/local dataPUL
Serial.println(dataPUL);     // Tampilkan hasil konversi ke Serial Monitor

 // Logika kontrol valve berdasarkan nilai pulsa
 // Jika pulsanya lebih dari 0 dan cekPulsaValve false
 if (cekPulsaValve == false && (dataPUL > 0 || dataPUL > 0.0)) {
  tampilLCD(String(dataPUL), NPku);        // Panggil fungsi tampilLCD()
  teganganVolt = analogRead(tegangan);     // Mengambil nilai tegangan
  Volt = ((teganganVolt * 0.00489) * 5);   // Menghitung nilai voltase

  // Jika voltasenya 0 atau kurang dari 5 maka panggil fungi valve_mati()
  if (Volt == 0.0 || Volt == 0 || Volt <= 5 || Volt <= 5.0) {
    valve_mati();     // Panggil fungsi valve_mati
  }else{
    // Jika voltasenya lebih dari 5 maka panggil fungsi valve_buka()
    valve_buka();             // Panggil fungsi valve_buka()
    cekPulsaValve = true;     // Atur nilai cekPulsaValve menjadi true
    cekPulsaValve2 = true;    // Atur nilai cekPulsaValve2 menjadi true
    setNilai = true;          // Atur nilai setNilai menjadi true
    setAwal  = true;          // Atur nilai setAwal menjadi true
    kirimHabis = false;       // Atur nilai kirimHabis menjadi false
    infoKedip = false;        // Atur nilai infoKedip menjadi false
    infoHabis = false;        // Atur nilai infoHabis menjadi false
  }
}
```

```
// Jika pulsanya lebih dari 0 dan cekPulsaValve2 bernilai false
if (cekPulsaValve2 == false && (dataPUL > 0 || dataPUL > 0.0)) {
  tampilLCD(String(dataPUL), NPku);        // Panggil fungsi tampilLCD()
  teganganVolt = analogRead(tegangan);     // Mengambil nilai tegangan
  Volt = ((teganganVolt * 0.00489) * 5);   // Menghitung nilai voltase

  // Jika voltasenya 0 atau kurang dari 5 maka panggil fungi valve_mati()
  if (Volt == 0.0 || Volt == 0 || Volt <= 5 || Volt <= 5.0) {
    valve_mati();     // Panggil fungsi valve_mati()
  }else{
    // Jika voltasenya lebih dari 5 maka panggil fungsi valve_buka()
    valve_buka();             // Panggil valve_buka
    cekPulsaValve = true;     // Atur nilai cekPulsaValve menjadi true
    cekPulsaValve2 = true;    // Atur nilai cekPulsaValve2 menjadi true
    setNilai = true;          // Atur nilai setNilai menjadi true
    setAwal  = true;          // Atur nilai setAwal menjadi true
    kirimHabis = false;       // Atur nilai kirimHabis menjadi false
    infoKedip = false;        // Atur nilai infoKedip menjadi false
    infoHabis = false;        // Atur nilai infoHabis menjadi false
  }
}

// Jika pulsanya lebih dari 0 dan cekPulsaValve bernilai true
if (cekPulsaValve == true && (dataPUL > 0 || dataPUL > 0.0)) {
  tampilLCD(String(dataPUL), NPku);        // Panggil fungsi tampilLCD()
  setNilai = true;
  setAwal  = true;
  kirimHabis = false;
  infoKedip = false;
  infoHabis = false;
}
// Jika pulsanya kurang dari 0 dan cekPulsaValve bernilai true
if (cekPulsaValve == true && (dataPUL < 0 || dataPUL < 0.0)) {
  valve_tutup();              // Panggil fungsi valve_tutup()
  cekPulsaValve = false;
  cekValveTutup = true;
}
// Jika pulsanya kurang dari 0 dan cekPulsaValve bernilai false
if(cekPulsaValve == false && (dataPUL < 0 || dataPUL < 0.0)){
  valve_mati();               // Panggil fungsi valve_mati()
  tampilLCD("0", NPku);       // Panggil fungsi tampilLCD()

  // Baca data pulsa baru dari Serial1
  String dataKU = Serial1.readString();
  dataKU.trim();
  Serial.println(dataKU);

  // Parsing data pelanggan dan pulsa dari dataKU
  // Ambil posisi awal dan akhir dari Nomor Pelanggan dalam string
  int indexSNP = dataKU.indexOf("#@");
  int indexLNP = dataKU.lastIndexOf("#&");
  String dataNP = dataKU.substring(indexSNP+2, indexLNP);
  NPku = dataNP;
```

```
      int indexSNM = dataKU.indexOf("#&");
      int indexLNM = dataKU.lastIndexOf("#!");
      String dataNM = dataKU.substring(indexSNM+2, indexLNM);

      int indexSPL = dataKU.indexOf("#!");
      int indexLPL = dataKU.lastIndexOf("##");
      String dataPL = dataKU.substring(indexSPL+2, indexLPL);

      dataPL.trim();                    // Nilai dataPL dibersihkan dari spasi
      dataPL.toFloat();                 // Nilai dataPL dikonversi ke tipe Float
      dataPUL = dataPL.toInt();         // Data pulsa dikonversi ke integer
      Serial.println(dataPUL);          // Tampilkan nilai dataPUL di serial monitor

      // Jika pulsa lebih dari 0, maka reset nilai cekPulsaValve2 dan cekValveTutup
      if(dataPUL > 0){
        cekPulsaValve2 = false;         // Atur nilai cekPulsaValve2 menjadi false
        cekValveTutup = false;          // Atur nilai cekValveTutup menjadi false
      }

      sendEsp("0", idMeter);     // Kirim status, dengan memanggil fungsi sendEsp()
    }
}

// Jika pesan kosong dan setNilai bernilai true
if(msg.equals("")){
  if(setNilai == true){
    setAwal = true;              // Atur nilai setAwal menjadi true, agar mulai proses awal
  }

}

// Jika setAwal bernilai true, maka jalankan fungsi berikut
if(setAwal == true){
  Serial.println("Ini muncul");
  String strdataPUL = String(dataPUL); // Tampilkan pulsa di LCD
  tampilLCD(strdataPUL, NPku);         // Panggil fungsi tampilLCD

  currentTime = millis();              // Hitung waktu sekarang
  // Cek apakah sudah lewat 1000 ms (1 detik) sejak pengukuran terakhir
  if (currentTime - startTime > 1000) {
     // Hitung debit aliran air dalam liter/menit
     // Rumus: flowRate (L/min) = pulseCount / 7.5 (berdasarkan karakteristik sensor)
    flowRate = pulseCount / 7.5;
     // Hitung volume air dalam liter selama 1 detik (karena 1 detik = 1/60 menit)
    flowLitres = (flowRate / 60);
     // Hitung volume air dalam mililiter
    flowMilliLitres = (flowRate / 60) * 1000;
     // Akumulasi total air yang sudah mengalir dalam mililiter
    totalMilliLitres += flowMilliLitres;

    // Tampilkan debit dan total air
    Serial.print("Debit air: ");
    Serial.print(flowRate);                    // dalam liter/menit
```

```
  Serial.print(" L/min\tTotal air: ");
  Serial.println(totalMilliLitres);          // total dalam mL

  pulseCount = 0;       // Reset jumlah pulsa dari sensor flow untuk periode berikutnya
  startTime = currentTime;     // Set waktu awal periode berikutnya

  pemakaian += flowRate; // Tambahkan pemakaian air (dalam L/min) ke total pemakaian
  dataPUL -= flowRate;  // Kurangi jumlah pulsa/kuota air sesuai debit air yang terpakai
}
 // Simpan waktu sekarang untuk pengukuran berikutnya (di luar if agar tetap diupdate)
cloopTime = currentTime;

// Tampilkan nilai air dalam mililiter yang mengalir saat ini
Serial.print(flowMilliLitres);
Serial.println(" XL");
// Tampilkan total air yang sudah digunakan (pemakaian)
Serial.print(pemakaian);
Serial.println(" K");


// Konversi nilai `pemakaian` ke bentuk string agar bisa digunakan
 // dalam pengiriman data serial, LCD, atau lainnya
String strPakai = String(pemakaian);

// Periksa kondisi untuk buzzer, dengan mengacu pada data pulsa (dataPUL)
if(dataPUL < 3000 && dataPUL > 500){
  buzzerKedip();               // Panggil fungsi buzzerKedip()
}
if(dataPUL <= 500 && dataPUL > 0 && buzzerTerusan == false){
  buzzerTerus();               // Panggil fungsi buzzerTerus()
  buzzerTerusan = true;        // Atur nilai buzzerTerusan menjadi true
  delay(5000);                 // Delay dengan nilai tertentu
}
if(dataPUL <= 500 && dataPUL > 0 && buzzerTerusan == true){
  buzzerMati();                // Panggil fungsi buzzerMati()
}

// Jika pulsa habis dan datanya belum dikirim ke ESP
if(dataPUL <= 0 && kirimHabis == false){
  if(pemakaian > 0){
    // Jika pemakaian lebih dari 0, maka panggil fungsi sendEsp()
    sendEsp(strPakai, idMeter);
  }else{
    // Jika kurang atau sama dengan 0, maka panggil fungsi sendEsp()
    // dengan mengirim nilai 0
    String nolPulsa = String(0);
    sendEsp(nolPulsa, idMeter);
  }
  valve_tutup();               // Panggil fungsi valve_tutup()
  tampilLCD("0", NPku);        // Panggil fungsi tampilLCD()
  cekPulsaValve = false;       // Atur nilai cekPulsaValve menjadi false
  kirimHabis = true;           // Atur kirimHabis menjadi true
  cekValveTutup = true;        // Atur cekValveTutup menjadi true
}
```

```
    // Jika pulsa habis dan data sudah dikirim, pastikan valve tertutup
    if(dataPUL <= 0 && kirimHabis == true){
      if(cekPulsaValve == true){
        valve_tutup();              // Panggil fungsi valve_tutup()
        cekPulsaValve = false;      // Atur nilai cekPulsaValve menjadi false
        cekValveTutup = true;       // Atur nilai cekValveTutup menjadi true
      }
      if(pemakaian > 0){            // Jika pemakaian lebih dari 0
        sendEsp(strPakai, idMeter);       // Panggil fungsi sendEsp() untuk kirim data
      }else{
        String nolPulsa = String(0);      // Atur nilai nolPulsa menjadi "0"
        sendEsp(nolPulsa, idMeter);       // Panggil fungsi sendEsp() untuk kirim data
      }
      valve_mati();                       // Panggil fungsi valve_mati()
      cekPulsaValve = false;              // Atur nilai cekPulsaValve menjadi false
      tampilLCD("0", NPku);               // Panggil fungsi tampilLCD()

      String dataKU = Serial1.readString();
      dataKU.trim();                      // Bersihkan nilai dataKU dengan menghapus spasi
      Serial.println(dataKU);
      int indexSNP = dataKU.indexOf("#@");
      int indexLNP = dataKU.lastIndexOf("#&");
      String dataNP = dataKU.substring(indexSNP+2, indexLNP);
      NPku = dataNP;

      int indexSNM = dataKU.indexOf("#&");
      int indexLNM = dataKU.lastIndexOf("#!");
      String dataNM = dataKU.substring(indexSNM+2, indexLNM);

      int indexSPL = dataKU.indexOf("#!");
      int indexLPL = dataKU.lastIndexOf("##");
      String dataPL = dataKU.substring(indexSPL+2, indexLPL);

      dataPL.trim();                // Bersihkan nilai dataPL dengan menghapus spasi
      dataPL.toFloat();             // Konversi tipe dataPL menjadi Float
      dataPUL = dataPL.toInt();     // Atur nilai dataPUL dengan nilai dataPL
      Serial.println(dataPUL);      // Menampilkan dataPUL di serial monitor

      // Jika nila data pulsa (dataPUL) lebih dari 0
      if(dataPUL > 0){
        cekPulsaValve2 = false;     // Atur nilai cekPulsaValve2 menjadi false
        cekValveTutup = false;      // Atur nilai cekValveTutup menjadi false
      }
    }
    // Jika pemakaian lebih dari 500, dan flowMilliLitres bernilai 0 (berhenti)
    if(pemakaian > 500 && flowMilliLitres == 0){
      sendEsp(strPakai, idMeter); // Panggil fungsi sendEsp()
    }
  }
}
```

```
// Fungsi jarakTutup()
// ------------------------------------------
// Fungsi ini digunakan untuk memantau kondisi pintu (terbuka atau tertutup)
// menggunakan sensor ultrasonic.
// Jika jarak melebihi ambang batas (jarakToleransi), pintu dianggap terbuka,
// dan sistem akan menutup valve serta menyalakan buzzer sebagai peringatan.
// Jika jarak berada dalam ambang batas, pintu dianggap tertutup,
// dan sistem akan membuka valve jika sebelumnya ditutup oleh kondisi pintu.
//
// Fungsi ini juga mengirimkan status pintu ("terbuka" / "tertutup") ke ESP
// melalui fungsi `sendPintu()` untuk keperluan pemantauan jarak jauh.
//
// Komponen yang digunakan:
// - Sensor ultrasonik (trigPin & echoPin)
// - Buzzer untuk peringatan
// - Valve untuk mengontrol aliran air
// - Serial komunikasi dengan ESP (idMeter)
// ------------------------------------------
void jarakTutup() {
  // Kirim sinyal LOW ke pin trig sensor ultrasonik selama 2 mikrodetik untuk reset
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  // Kirim sinyal HIGH ke pin trig selama 10 mikrodetik untuk memulai pengukuran jarak
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  // Baca durasi pantulan dari pin echo (dalam mikrodetik)
  duration = pulseIn(echoPin, HIGH);
  // Hitung jarak berdasarkan kecepatan suara (0.034 cm/us)
  // dan dibagi 2 karena gelombang pergi-pulang
  distance = duration * 0.034 / 2;

//  Serial.print(distance);
//  Serial.println(" cm");

  // ===== LOGIKA KETIKA JARAK MELEBIHI BATAS TOLERANSI (PINTU TERBUKA) =====
  if (distance > jarakToleransi) {
    buzzerKedip();   // Aktifkan buzzer sebagai alarm
     // Jika sebelumnya belum terdeteksi pintu terbuka dan valve belum ditutup
    if(cekPintu == false && cekValveTutup == false){
      valve_tutup();        // Panggil fungsi valve_tutup()
      cekPintu = true;      // Tandai bahwa pintu sudah terbuka
      // TambahanPintu
      sendPintu("terbuka", idMeter);      // Kirim status pintu ke server
    }
    // Jika valve sudah ditutup, pastikan valve tetap dalam keadaan mati (tidak aktif)
    if(cekPintu == false && cekValveTutup == true){
      valve_mati();         // Panggil fungsi valve_mati()
      cekPintu = true;      // Tandai bahwa pintu sudah terbuka
    }
    if(cekPintu == true){   // Jika sudah terdeteksi pintu terbuka
      valve_mati();         // Panggil valve_mati()
    }
//    Serial.println("Kebuka");
```

```
  } else {    // ===== LOGIKA KETIKA PINTU DALAM BATAS NORMAL (TERTUTUP) =====
    buzzerMati();    // Panggil fungsi buzzerMati()
     // Jika sebelumnya pintu terbuka dan valve belum ditutup
    if(cekPintu == true && cekValveTutup == false){
      valve_buka();         // Panggil fungsi valve_buka()
      cekPintu = false;     // Tandai bahwa pintu sudah ditutup
      // TambahanPintu
      sendPintu("tertutup", idMeter);     // Kirim status pintu tertutup ke server
    }
    // Jika valve sudah ditutup, pastikan tetap tidak aktif
    if(cekPintu == true && cekValveTutup == true){
      valve_mati();         // Panggil fungsi valve_mati()
      cekPintu = true;
    }
    // Jika pintu memang sudah tertutup sebelumnya, tetap jaga valve mati
    if(cekPintu == false){
      valve_mati();         // Panggil fungsi valve_mati()
    }
    cekPintu = false;
  }
}

// Fungsi cekMiring()
// ---------------------------------------------------
// Fungsi ini memantau kondisi kemiringan perangkat
// menggunakan sensor tilt (switch atau sensor kemiringan digital).
// Jika sensor membaca nilai LOW (0), artinya perangkat miring,
// dan buzzer akan berbunyi terus-menerus sebagai peringatan.
// Jika perangkat dalam kondisi normal (sensor HIGH), buzzer dimatikan.
//
// Fungsi ini berguna untuk keamanan, misalnya ketika perangkat jatuh,
// miring karena gangguan, atau disengaja digeser dari posisi normal.
//
// Delay 1 detik digunakan untuk menghindari pembacaan yang terlalu sering.
// ---------------------------------------------------
void cekMiring() {
  // Baca nilai dari pin sensor tilt (sensor kemiringan)
  int bacaSensor = digitalRead(miringPin);
  // Tampilkan nilai pembacaan ke Serial Monitor (0 atau 1)
  Serial.println(bacaSensor);
  // Jika sensor mendeteksi kemiringan (nilai 0 menandakan tilt/miring)
  if (bacaSensor == 0) {
    Serial.println("Tilt detected");      // Tampilkan pesan deteksi kemiringan
    buzzerTerus();                       // Aktifkan buzzer secara terus-menerus
  } else {
    Serial.println("Normal");            // Tampilkan status normal jika tidak miring
    buzzerMati();                        // Matikan buzzer
  }
  delay(1000);                           // Delay 1 detik sebelum pembacaan berikutnya
}
```

```
// Fungsi buzzerKedip()
// -------------------------------------------------
// Mengaktifkan buzzer dengan bunyi putus-putus (kedip).
// Digunakan sebagai peringatan ringan, misalnya ketika pulsa hampir habis atau
// dalam kondisi tertentu yang menyebabkan fungsi buzzerKedip() dipanggil.
// -------------------------------------------------
void buzzerKedip(){
  tone(buzzerPin, 500);     // Mengaktifkan buzzer dengan frekuensi 500 Hz
  delay(1000);              // Delay 1 detik selama buzzer berbunyi
  noTone(buzzerPin);        // Mematikan suara buzzer
  delay(1000);              // Delay 1 detik selama buzzer tidak berbunyi
}

// Fungsi buzzerTerus()
// -------------------------------------------------
// Mengaktifkan buzzer dengan bunyi terus menerus selama 4 detik.
// Digunakan sebagai peringatan keras, misalnya ketika pulsa habis atau
// dalam kondisi tertentu yang menyebabkan fungsi buzzerTerus() dipanggil.
// -------------------------------------------------
void buzzerTerus(){
  tone(buzzerPin, 500);     // Mengaktifkan buzzer dengan frekuensi 500 Hz
  delay(4000);              // Delay 4 detik selama buzzer berbunyi
}

// Fungsi buzzerMati()
// -------------------------------------------------
// Mematikan buzzer dengan menghentikan nada pada pin buzzer.
// Dipanggil saat sistem kembali dalam kondisi normal.
// -------------------------------------------------
void buzzerMati(){
  noTone(buzzerPin);        // Buzzer tidak berbunyi
}

// Fungsi cekTegangan()
// -------------------------------------------------
// Fungsi ini membaca nilai tegangan menggunakan input analog
// dan mengonversinya menjadi satuan volt.
// Jika tegangan di bawah atau sama dengan 5V, maka sistem akan
// menutup valve sebagai langkah pengamanan.
// Jika tegangan kembali normal (>= 5V), valve akan dibuka kembali.
//
// Variabel flag `nivolt` dan `nivoltbuka` digunakan untuk
// mencegah pengulangan aksi yang sama secara terus-menerus.
// Fungsi ini penting untuk mendeteksi kondisi low-voltage
// yang bisa menyebabkan sistem tidak stabil.
// -------------------------------------------------
void cekTegangan() {
  teganganVolt = analogRead(tegangan);    // Baca nilai analog dari pin tegangan
  Volt = ((teganganVolt * 0.00489) * 5); // Konversi nilai analog (0–1023) ke volt
  //  Serial.print("TeganganX : ");
  //  Serial.println(Volt);

  // Jika tegangan terlalu rendah (<= 5V)
  if ((Volt == 0.0 || Volt == 0 || Volt <= 5 || Volt <= 5.0)) {
```

```
    // Jika belum dalam kondisi tegangan rendah, tutup valve
    if (nivolt == false) {
      valve_tutup(); // Panggil fungsi valve_tutup()
      nivolt = true;
      nivoltbuka = false;
    } else {         // Jika sudah dalam kondisi tegangan rendah, cukup matikan valve
      valve_mati();  // Panggil fungsi valve_mati()
    }
  // Jika tegangan kembali normal (> 5V) dan sebelumnya dalam kondisi tegangan rendah
  } else if (nivolt == true && nivoltbuka == false && (Volt >= 5 || Volt >= 5.0)) {
    valve_buka();    // Panggil fungsi valve_buka()
    nivolt = false;
    nivoltbuka = true;
  } else { // Jika kondisi normal tanpa perubahan status, cukup matikan valve
    valve_mati();    // Panggil fungsi valve_mati()
  }
}

// Fungsi tampilLCD()
// -------------------------------------------------
// Menampilkan informasi ID pelanggan dan saldo pulsa air
// ke layar LCD PCD8544. Informasi yang ditampilkan:
// - Baris 1: ID Pelanggan
// - Baris 2: Pulsa dalam Rupiah
// - Baris 3: Estimasi liter air yang bisa digunakan berdasarkan pulsa
// Konversi pulsa ke liter menggunakan rumus: liter = pulsa / 3.4
// -------------------------------------------------
void tampilLCD(String lcdpulsa, String nopelanggan) {
  lcd.clear();                          // Bersihkan LCD
  lcd.setCursor(0, 0);                  // Atur titik untuk menampilkan nilai
  lcd.print("ID:" + nopelanggan);       // Tampilkan nilai
  lcd.setCursor(0, 1);                  // Atur titik untuk menampilkan nilai
  lcd.print("P :Rp " + lcdpulsa);       // Tampilkan nilai
  // lcd.setCursor(0, 3);
  // lcd.print("U : usage");
  lcd.setCursor(0, 2);                  // Atur titik untuk menampilkan nilai
  float pulsaL = lcdpulsa.toFloat();    // Atur nilai pulsaL dari lcdpulsa
  float hasilL = pulsaL/3.4;            // Hitung hasilnya
  lcd.print("(" + String(hasilL) +" L)");// Tampilkan nilai
}
```

```
// Fungsi sendEsp()
// ------------------------------------------------
// Mengirim data pulsa dan ID pelanggan melalui komunikasi serial (Serial, Serial1, dan myArd).
// Format pengiriman data:
// - "SendData"
// - Pulsa diikuti tanda '#' (contoh: 12000#)
// - ID diikuti tanda '+' (contoh: 12345678+)
// Setelah mengirim data, pemakaian air disetel ulang ke nol.
// Digunakan untuk sinkronisasi data dengan modul ESP atau Arduino lain.
// ------------------------------------------------
void sendEsp(String sendPulsa, String sendID) {
  Serial.println("SendData");
  Serial1.println("SendData");
  myArd.println("SendData");
  Serial.println(sendPulsa + "#");
  Serial1.println(sendPulsa + "#");
  myArd.println(sendPulsa + "#");
  Serial.println(sendID + "+");
  Serial1.println(sendID + "+");
  myArd.println(sendID) + "+";
  pemakaian = 0;
}

// Fungsi sendPintu()
// ------------------------------------------------
// Mengirim status pintu dan ID pelanggan melalui komunikasi serial,
// dengan format serupa seperti sendEsp(). Status bisa berupa
// "terbuka" atau "tertutup".
// Digunakan saat pintu penutup meteran air terdeteksi berubah status.
// ------------------------------------------------
void sendPintu(String sendPintu, String sendID) {
  Serial.println("SendPintu");
  Serial1.println("SendPintu");
  myArd.println("SendPintu");
  Serial.println(sendPintu + "#");
  Serial1.println(sendPintu + "#");
  myArd.println(sendPintu + "#");
  Serial.println(sendID + "+");
  Serial1.println(sendID + "+");
  myArd.println(sendID) + "+";
  pemakaian = 0;
}

// Fungsi sendBeep()
// ------------------------------------------------
// Mengirim perintah suara buzzer (beep) dan ID pelanggan melalui serial,
// biasanya digunakan sebagai bentuk peringatan yang dapat dikendalikan
// dari sistem eksternal.
// Format data sama seperti fungsi lainnya (data# dan ID+).
// ------------------------------------------------
void sendBeep(String dataBeep, String dataIDBeep) {
  Serial.println("SendBeep");
  Serial1.println("SendBeep");
  myArd.println("SendBeep");
```

```
  Serial.println(dataBeep + "#");
  Serial1.println(dataBeep + "#");
  myArd.println(dataBeep + "#");
  Serial.println(dataIDBeep + "+");
  Serial1.println(dataIDBeep + "+");
  myArd.println(dataIDBeep) + "+";
}

// Fungsi valve_buka()
// ------------------------------------------------
// Mengaktifkan katup (valve) agar terbuka selama 4.7 detik
// dengan mengatur arah arus motor valve.
// Setelah waktu buka selesai, katup dimatikan (motor stop).
// ------------------------------------------------
void valve_buka() {
  // Set pinValve8 ke LOW (0V)
  digitalWrite(pinValve8, LOW);
  // Set pinValve9 ke HIGH (5V), sehingga terjadi aliran arus satu arah
  digitalWrite(pinValve9, HIGH);
  // Tunggu selama 4700 milidetik (4,7 detik) untuk memastikan valve terbuka sepenuhnya
  delay(4700);
  // Setelah selesai membuka, hentikan aliran listrik ke valve
  // dengan memanggil fungsi valve_mati()
  valve_mati();
}

// Fungsi valve_tutup()
// ------------------------------------------------
// Mengaktifkan katup (valve) agar menutup selama 5 detik
// dengan arah arus motor sebaliknya dari valve_buka().
// Setelah itu, katup dimatikan untuk menghentikan motor.
// ------------------------------------------------
void valve_tutup() {
  // Set pinValve8 ke HIGH (5V), sehingga terjadi aliran arus satu arah
  digitalWrite(pinValve8, HIGH);
  // Set pinValve9 ke LOW (0V)
  digitalWrite(pinValve9, LOW);
  delay(5000);
  valve_mati();
}

// Fungsi valve_mati()
// ------------------------------------------------
// Mematikan semua arus ke motor valve, memastikan tidak ada
// pergerakan atau konsumsi daya pada motor valve.
// ------------------------------------------------
void valve_mati() {
  digitalWrite(pinValve8, LOW); // Set pinValve8 ke LOW (0V)
  digitalWrite(pinValve9, LOW); // Set pinValve9 ke LOW (0V)
}
```