**CODE 1:**

**VOLUME 1:**

#include <Wire.h>

#include <LiquidCrystal\_I2C.h>

volatile int flow\_frequency1; // Measures flow sensor 1 pulses

volatile int flow\_frequency2; // Measures flow sensor 2 pulses

float vol1 = 0.0, l\_minute1;

float vol2 = 0.0, l\_minute2;

unsigned char flowsensor1 = 2; // Sensor 1 Input

unsigned char flowsensor2 = 3; // Sensor 2 Input

unsigned long currentTime;

unsigned long cloopTime;

unsigned long lastResetTime = 0;

const unsigned long volumeResetInterval = 86400000; // 24 hours in milliseconds

// I2C LCD address (change this if your address is different)

#define I2C\_LCD\_ADDR 0x27

LiquidCrystal\_I2C lcd(I2C\_LCD\_ADDR, 16, 2); // Initialize the I2C LCD

void flow1() // Interrupt function for sensor 1

{

flow\_frequency1++;

}

void flow2() // Interrupt function for sensor 2

{

flow\_frequency2++;

}

void setup()

{

pinMode(flowsensor1, INPUT);

pinMode(flowsensor2, INPUT);

digitalWrite(flowsensor1, HIGH); // Optional Internal Pull-Up

digitalWrite(flowsensor2, HIGH); // Optional Internal Pull-Up

Serial.begin(9600);

lcd.init(); // initialize the lcd

lcd.backlight();

lcd.begin(16, 2);

24

lcd.print("Water Flow Meter");

currentTime = millis();

cloopTime = currentTime;

lastResetTime = currentTime; // Set the initial reset time to the current time

attachInterrupt(digitalPinToInterrupt(flowsensor1), flow1, RISING); // Setup Interrupt for sensor 1

attachInterrupt(digitalPinToInterrupt(flowsensor2), flow2, RISING); // Setup Interrupt for sensor 2

}

void loop()

{

currentTime = millis();

// Check if 24 hours have passed since the last volume reset

if (currentTime - lastResetTime >= volumeResetInterval)

{

lastResetTime = currentTime; // Update the last reset time

// Reset the volume variables to zero

vol1 = 0.0;

vol2 = 0.0;

}

// Every second, calculate and print litres/hour for sensor 1

if (currentTime >= (cloopTime + 1000))

{

cloopTime = currentTime; // Updates cloopTime

bool hasFlow1 = (flow\_frequency1 != 0);

bool hasFlow2 = (flow\_frequency2 != 0);

if (hasFlow1)

{

l\_minute1 = (flow\_frequency1 / 7.5); // (Pulse frequency x 60 min) / 7.5Q = flowrate in L/hour

lcd.clear();

lcd.setCursor(0, 0);

lcd.print("S1 flow: ");

lcd.print(l\_minute1);

lcd.print(" L/M");

l\_minute1 = l\_minute1 / 60;

lcd.setCursor(0, 1);

vol1 = vol1 + l\_minute1;

lcd.print("Vol 1: ");

lcd.print(vol1);

25

lcd.print(" L");

flow\_frequency1 = 0; // Reset Counter for sensor 1

Serial.print("Sensor 1: ");

Serial.print(l\_minute1, DEC); // Print litres/hour

Serial.println(" L/Sec");

}

else

{

// Print Volume 1 on row 1 since there is no flow in sensor 1

lcd.clear();

lcd.setCursor(0, 0);

lcd.print("Volume 1: ");

lcd.print(vol1);

lcd.print(" L");

}

if (hasFlow2)

{

l\_minute2 = (flow\_frequency2 / 7.5); // (Pulse frequency x 60 min) / 7.5Q = flowrate in L/hour

lcd.setCursor(0, 0);

lcd.print("S2 flow: ");

lcd.print(l\_minute2);

lcd.print(" L/M");

l\_minute2 = l\_minute2 / 60;

lcd.setCursor(0, 1);

vol2 = vol2 + l\_minute2;

lcd.print("Vol 2: ");

lcd.print(vol2);

lcd.print(" L");

flow\_frequency2 = 0; // Reset Counter for sensor 2

Serial.print("Sensor 2: ");

Serial.print(l\_minute2, DEC); // Print litres/hour

Serial.println(" L/Sec");

}

else

{

// Print Volume 2 on row 2 since there is no flow in sensor 2

lcd.setCursor(0, 1);

lcd.print("Volume 2: ");

lcd.print(vol2);

lcd.print(" L");

}

26

if (!hasFlow1 && !hasFlow2)

{

// Both sensors have no flow, print "Volume 1" on row 1 and "Volume 2" on row 2 lcd.setCursor(0, 0);

lcd.print("Volume 1");

lcd.setCursor(0, 1);

lcd.print("Volume 2");

}

// Inside the loop() function

if (hasFlow1 && hasFlow2)

{

// Both sensors have flow, print "Volume 1" on row 1 and "Volume 2" on row 2 lcd.setCursor(0, 0);

lcd.print("Volume 1");

lcd.setCursor(0, 1);

lcd.print("Volume 2");

}

}

}

**CODE 2:**

**VOLUME:2**

#include <Wire.h>

#include <LiquidCrystal\_I2C.h>

volatile int flow\_frequency1; // Measures flow sensor 1 pulses

volatile int flow\_frequency2; // Measures flow sensor 2 pulses

float vol1 = 0.0, l\_minute1;

float vol2 = 0.0, l\_minute2;

unsigned char flowsensor1 = 2; // Sensor 1 Input

unsigned char flowsensor2 = 3; // Sensor 2 Input

// I2C LCD address (change this if your address is different)

#define I2C\_LCD\_ADDR 0x27

LiquidCrystal\_I2C lcd(I2C\_LCD\_ADDR, 16, 2); // Initialize the I2C LCD

unsigned long currentTime;

unsigned long cloopTime;

unsigned long lastResetTime = 0;

const unsigned long volumeResetInterval = 86400000; // 24 hours in milliseconds

void flow1() // Interrupt function for sensor 1

{

flow\_frequency1++;

}

27

void flow2() // Interrupt function for sensor 2

{

flow\_frequency2++;

}

void setup() {

pinMode(flowsensor1, INPUT);

pinMode(flowsensor2, INPUT);

digitalWrite(flowsensor1, HIGH); // Optional Internal Pull-Up

digitalWrite(flowsensor2, HIGH); // Optional Internal Pull-Up

Serial.begin(9600);

lcd.init(); // initialize the lcd

lcd.backlight();

lcd.begin(16, 2);

lcd.print("Water Flow Meter");

currentTime = millis();

cloopTime = currentTime;

lastResetTime = currentTime; // Set the initial reset time to the current time

attachInterrupt(digitalPinToInterrupt(flowsensor1), flow1, RISING); // Setup Interrupt for sensor 1

attachInterrupt(digitalPinToInterrupt(flowsensor2), flow2, RISING); // Setup Interrupt for sensor 2

}

void loop() {

currentTime = millis();

// Check if 24 hours have passed since the last volume reset

if (currentTime - lastResetTime >= volumeResetInterval) {

lastResetTime = currentTime; // Update the last reset time

// Reset the volume variables to zero

vol1 = 0.0;

vol2 = 0.0;

}

// Every second, calculate and print litres/hour for sensor 1

if (currentTime >= (cloopTime + 1000)) {

cloopTime = currentTime; // Updates cloopTime

bool hasFlow1 = (flow\_frequency1 != 0);

bool hasFlow2 = (flow\_frequency2 != 0);

lcd.clear();

if (hasFlow1 && hasFlow2) {

28

l\_minute1 = (flow\_frequency1 / 7.5); // (Pulse frequency x 60 min) / 7.5Q = flowrate in L/hour

l\_minute1 = l\_minute1 / 60;

vol1 = vol1 + l\_minute1;

l\_minute2 = (flow\_frequency2 / 7.5); // (Pulse frequency x 60 min) / 7.5Q = flowrate in L/hour

l\_minute2 = l\_minute2 / 60;

vol2 = vol2 + l\_minute2;

// If there's no flow from both sensors, display Volume 1 and Volume 2 lcd.setCursor(0, 0); // Set cursor to the first row

lcd.print("Volume 1: ");

lcd.print(vol1);

lcd.print(" L");

lcd.setCursor(0, 1); // Set cursor to the second row

lcd.print("Volume 2: ");

lcd.print(vol2);

lcd.print(" L");

}

else if (hasFlow1) {

l\_minute1 = (flow\_frequency1 / 7.5); // (Pulse frequency x 60 min) / 7.5Q = flowrate in L/hour

l\_minute1 = l\_minute1 / 60;

vol1 = vol1 + l\_minute1;

lcd.setCursor(0, 0); // Set cursor to the first row

lcd.print("S1 flow: ");

lcd.print(l\_minute1);

lcd.print(" L/M");

lcd.setCursor(0, 1); // Set cursor to the second row

lcd.print("Volume 1: ");

lcd.print(vol1);

lcd.print(" L");

} else if (hasFlow2) {

l\_minute2 = (flow\_frequency2 / 7.5); // (Pulse frequency x 60 min) / 7.5Q = flowrate in L/hour

l\_minute2 = l\_minute2 / 60;

vol2 = vol2 + l\_minute2;

lcd.setCursor(0, 0); // Set cursor to the first row

lcd.print("S2 flow: ");

lcd.print(l\_minute2);

lcd.print(" L/M");

lcd.setCursor(0, 1); // Set cursor to the second row

lcd.print("Volume 2: ");

lcd.print(vol2);

lcd.print(" L");

29

} else {

// If there's no flow from both sensors, display Volume 1 and Volume 2 lcd.setCursor(0, 0); // Set cursor to the first row

lcd.print("Volume 1: ");

lcd.print(vol1);

lcd.print(" L");

lcd.setCursor(0, 1); // Set cursor to the second row

lcd.print("Volume 2: "); b

lcd.print(vol2);

lcd.print(" L");

}

flow\_frequency1 = 0; // Reset Counter for sensor 1

flow\_frequency2 = 0; // Reset Counter for sensor 2

}

}

**CODE 3:**

**FINAL CODE**

#include <Wire.h>

#include <LiquidCrystal\_I2C.h>

#include <SoftwareSerial.h>

SoftwareSerial mySerial(8, 9); //(rx,tx)

volatile int flow\_frequency1; // Measures flow sensor 1 pulses volatile int flow\_frequency2;

int count = 0;

int mcount = 0;

int acount = 0;

// Measures flow sensor 2 pulses

float vol1 = 0.0, l\_minute1;

float vol2 = 0.0, l\_minute2;

float fvol = 0.0;

float f1vol = 0.0;

float mvol1 = 0.0, ml\_minute1;

float mvol2 = 0.0, ml\_minute2;

unsigned char flowsensor1 = 2; // Sensor 1 Input

unsigned char flowsensor2 = 3; // Sensor 2 Input

const int switchPin = 6; // For turning on and off the monitor mode const int relay = 7; //for relay-solenoid

const int beep = 5;

// I2C LCD address (change this if your address is different) #define I2C\_LCD\_ADDR 0x27

30

LiquidCrystal\_I2C lcd(I2C\_LCD\_ADDR, 16, 2); // Initialize the I2C LCD

unsigned long currentTime;

unsigned long cloopTime;

unsigned long lastResetTime = 0;

const unsigned long volumeResetInterval = 86400000; // 24 hours in milliseconds bool monitorMode = false; // Indicates whether the monitor mode is active or not

void flow1() // Interrupt function for sensor 1

{

flow\_frequency1++;

}

void flow2() // Interrupt function for sensor 2

{

flow\_frequency2++;

}

void setup() {

pinMode(flowsensor1, INPUT);

pinMode(flowsensor2, INPUT);

pinMode(switchPin, INPUT\_PULLUP);

digitalWrite(flowsensor1, HIGH); // Optional Internal Pull-Up

digitalWrite(flowsensor2, HIGH); // Optional Internal Pull-Up

pinMode(relay, OUTPUT);

pinMode(beep, OUTPUT);

Serial.begin(9600);

mySerial.begin(9600);

// Set GSM module to use text mode for SMS

lcd.init(); // initialize the lcd

lcd.backlight();

lcd.begin(16, 2);

lcd.print("Water Flow Meter");

currentTime = millis();

cloopTime = currentTime;

lastResetTime = currentTime; // Set the initial reset time to the current time

attachInterrupt(digitalPinToInterrupt(flowsensor1), flow1, RISING); // Setup Interrupt for sensor 1 attachInterrupt(digitalPinToInterrupt(flowsensor2), flow2, RISING); // Setup Interrupt for sensor 2 }

void loop() {

currentTime = millis();

31

int switchState = digitalRead(switchPin); // Read the state of the switch

if (switchState == LOW) {

Serial.println("Switch is pressed");

monitorMode = true;

// Activate monitor mode

} else {

monitorMode = false; // Deactivate monitor mode

}

// If monitor mode is active, display "Monitoring Leakage"

if (monitorMode) {

lcd.clear();

lcd.setCursor(0, 0); // Set cursor to the first row

lcd.print("Monitoring");

lcd.setCursor(0, 1); // Set cursor to the second row

lcd.print("Leakage...");

ml\_minute1 = (flow\_frequency1 / 7.045); // (Pulse frequency x 60 min) / 7.5Q = flowrate in L/hour ml\_minute1 = ml\_minute1 / 60;

mvol1 = mvol1 + ml\_minute1;

ml\_minute2 = (flow\_frequency2 / 7.045); // (Pulse frequency x 60 min) / 7.5Q = flowrate in L/hour ml\_minute2 = ml\_minute2 / 60;

mvol2 = mvol2 + ml\_minute2;

fvol = mvol1 + mvol2;

flow\_frequency1 = 0;

flow\_frequency2 = 0;

Serial.println(mvol1);

Serial.println(mvol2);

Serial.println(fvol);

// Check if fvol crosses 2 liters

if (fvol >= 2.0) {

lcd.clear();

lcd.setCursor(0, 0); // Set cursor to the first row

lcd.print("Leak Detected");

lcd.setCursor(0, 1); // Set cursor to the second row

lcd.print("Alarming");

//code for sending noti regarding leak

if (count < 1) {

sendSMS("Leak detected - ring the GSM for 5 secs to stop water flow");

count = 99;

}

32

delay(5000);

checkIncomingCall();

}

checkIncomingCall();

} else {

// Check if 24 hours have passed since the last volume reset

if (currentTime - lastResetTime >= volumeResetInterval) {

lastResetTime = currentTime; // Update the last reset time

// Reset the volume variables to zero

vol1 = 0.0;

vol2 = 0.0;

f1vol = 0.0;

count = 0;

fvol = 0;

mvol2 = 0;

mvol1 = 0;

acount = 0;

}

count = 0;

fvol = 0;

mvol2 = 0;

mvol1 = 0;

acount = 0;

// Every second, calculate and print litres/hour for sensor 1

if (currentTime >= (cloopTime + 1000)) {

cloopTime = currentTime; // Updates cloopTime

bool hasFlow1 = (flow\_frequency1 != 0);

bool hasFlow2 = (flow\_frequency2 != 0);

lcd.clear();

if (hasFlow1 && hasFlow2) {

l\_minute1 = (flow\_frequency1 / 7.045); // (Pulse frequency x 60 min) / 7.5Q = flowrate in L/hour l\_minute1 = l\_minute1 / 60;

vol1 = vol1 + l\_minute1;

l\_minute2 = (flow\_frequency2 / 7.045); // (Pulse frequency x 60 min) / 7.5Q = flowrate in L/hour l\_minute2 = l\_minute2 / 60;

vol2 = vol2 + l\_minute2;

lcd.setCursor(0, 0); // Set cursor to the first row

lcd.print("Volume 1: ");

lcd.print(vol1);

lcd.print(" L");

lcd.setCursor(0, 1); // Set cursor to the second row

lcd.print("Volume 2: ");

33

lcd.print(vol2);

lcd.print(" L");

f1vol = vol1 + vol2;

} else if (hasFlow1) {

l\_minute1 = (flow\_frequency1 / 7.045); // (Pulse frequency x 60 min) / 7.5Q = flowrate in L/hour l\_minute1 = l\_minute1 / 60;

vol1 = vol1 + l\_minute1;

lcd.setCursor(0, 0); // Set cursor to the first row

lcd.print("S1 flow: ");

lcd.print(l\_minute1);

lcd.print(" L/M");

lcd.setCursor(0, 1); // Set cursor to the second row

lcd.print("Volume 1: ");

lcd.print(vol1);

lcd.print(" L");

f1vol = vol1;

} else if (hasFlow2) {

l\_minute2 = (flow\_frequency2 / 7.045); // (Pulse frequency x 60 min) / 7.5Q = flowrate in L/hour l\_minute2 = l\_minute2 / 60;

vol2 = vol2 + l\_minute2;

lcd.setCursor(0, 0); // Set cursor to the first row

lcd.print("S2 flow: ");

lcd.print(l\_minute2);

lcd.print(" L/M");

lcd.setCursor(0, 1); // Set cursor to the second row

lcd.print("Volume 2: ");

lcd.print(vol2);

lcd.print(" L");

f1vol = vol2;

} else {

// If there's no flow from both sensors, display Volume 1 and Volume 2

lcd.setCursor(0, 0); // Set cursor to the first row

lcd.print("Volume 1: ");

lcd.print(vol1);

lcd.print(" L");

lcd.setCursor(0, 1); // Set cursor to the second row

lcd.print("Volume 2: ");

lcd.print(vol2);

lcd.print(" L");

f1vol = vol1 + vol2;

}

Serial.println(f1vol);

flow\_frequency1 = 0; // Reset Counter for sensor 1

flow\_frequency2 = 0; // Reset Counter for sensor 2

34

if (f1vol >= 5) {

lcd.clear();

lcd.setCursor(0, 0); // Set cursor to the first row

lcd.print("LIMIT REACHED!");

lcd.setCursor(0, 1); // Set cursor to the second row

lcd.print("TOTAL VOL: ");

lcd.print(f1vol);

lcd.print(" L");

alarm();

acount=99;

}

}

}

}

void updateSerial() {

delay(500);

while (Serial.available()) {

mySerial.write(Serial.read()); // Forward what Serial received to Software Serial Port }

while (mySerial.available()) {

Serial.write(mySerial.read()); // Forward what Software Serial received to Serial Port }

}

void sendSMS(const String& message) {

// Your SMS sending code goes here

mySerial.println("AT");

updateSerial();

mySerial.println("AT+CMGF=1");

updateSerial();

mySerial.println("AT+CMGS=\"+917695887731\""); // enter your phone number here (prefix country code) updateSerial();

mySerial.print(message); // Pass the SMS message as a parameter

updateSerial();

mySerial.write(26);

delay(5000);

}

35

void checkIncomingCall() {

if (mySerial.available()) {

String response = mySerial.readStringUntil('\n');

Serial.println(response);

// If the incoming call notification is detected

if (response.indexOf("RING") != -1) {

// You can add your custom logic here to handle the incoming call

Serial.println("Incoming call detected!");

// Turn on the relay

digitalWrite(relay, HIGH);

// Send an SMS

sendSMS("Water flow has been stopped. Check after going home.");

// For example, you can hang up the call after a few seconds

delay(2000); // Wait for 2 seconds

mySerial.println("ATH"); // Hang up the call

Serial.println("Call hung up.");

}

}

}

void alarm() {

if(acount<1){

sendSMS("water usage limit reached.");

}

digitalWrite(beep, HIGH);

delay(1000);

digitalWrite(beep, LOW);

delay(5000);

}