## Code Implementation

Top functions are mentioned with line number



```
// Include LCD display library for I2C
#include <LiquidCrystal_I2C.h>
// Include NewPing Library
// Include NewPing Library
#include "NewPing.h"

#include <Keypad.h>
// Hook up HC-SR04 with Trig to Arduino
#define TRIGGER_PIN 11
#define ECHO_PIN 12


// Maximum distance we want to ping for (in mm).
#define MAX_DISTANCE 252

// NewPing setup of pins and maximum distance.
NewPing sonar(TRIGGER_PIN, ECHO_PIN, MAX_DISTANCE);

// Eta Low Level Trigger
// Password Length
const int Password_Length = 7;  // ->>>> Password And Length -> Changing
// Character to hold password input
String Data;
// Password
String Master = "2007111";
// ->>>> Here is the password

//   # for Turn Off and * for Password Reset

// Motor A connections
int pumpMotorOutput = 10;

bool flag = true;
bool lock_state = false;  // true means locked and false means unlocked
// Pin connected to lock relay signal
int gateValveOutput = 13;

// Counter for character entries
byte data_count = 0;

// Character to hold key input
char customKey;

// Constants for row and column sizes
const byte ROWS = 4;
const byte COLS = 4;

// Array to represent keys on keypad
char hexaKeys[ROWS][COLS] = {
  { '1', '2', '3', 'A' },
  { '4', '5', '6', 'B' },
  { '7', '8', '9', 'C' },
  { '*', '0', '#', 'D' }
};

// Connections to Arduino
byte rowPins[ROWS] = { 9, 8, 7, 6 };
byte colPins[COLS] = { 5, 4, 3, 2 };


// Create keypad object
Keypad customKeypad = Keypad(makeKeymap(hexaKeys), rowPins, colPins, ROWS, COLS);

// Create LCD object : Use 0x27 If 0x27 Doesn't work
LiquidCrystal_I2C lcd(0x27, 16, 2);

```

```
//Water level
int water1 = A0;
int water2 = A1;
int water3 = A2;
int val1 = 0, val2 = 0, val3 = 0, total = 0;
int level = 0;
//

void setup() {
  // Setup LCD with backlight and initialize
  lcd.init();
  lcd.backlight();
  lcd.print("Enter Password:");

  // Set gateValveOutput as an OUTPUT pin
  pinMode(gateValveOutput, OUTPUT);
  digitalWrite(gateValveOutput, HIGH);

  pinMode(pumpMotorOutput, OUTPUT);

  //start the serial monitor
  Serial.begin(9600);
}

void loop() {

  // Initialize LCD and print
  //Serial.println((sonar.ping_median(5) / 2) * 0.343);
  lcd.setCursor(0, 0);
  lcd.print("Enter Password:");

  // Look for keypress
  customKey = customKeypad.getKey();

  if (customKey) {

    // Enter keypress into array and increment counter
    Data += customKey;
    lcd.setCursor(data_count, 1);
    lcd.print(Data[data_count]);

    if (flag) {
      flag = false;
      Serial.print("Your Password : ");
    }
    Serial.print(Data[data_count]);
    data_count++;


    if (customKey == '*') {
      Data = "";
      data_count = 0;
      Serial.print("\n Password Cleaned !! \n");
      flag = true;

      lcd.clear();
      clearData();

      // Incorrect Password
      lcd.print("Pin Cleaned");
      delay(1000);
    }
    if (customKey == '#') {
      Data = "";
      data_count = 0;
      Serial.print("\n Password Cleaned !! \n");
```

```
      Serial.print("Power OFF Now !!");
      flag = true;

      digitalWrite(gateValveOutput, HIGH);

      lock_state = true;

      lcd.clear();
      clearData();

      // Incorrect Password
      lcd.print("Lock Btn Press");
      delay(1000);
    }
  }

  // See if we have reached the password length
  if (data_count == Password_Length) {

    lcd.clear();

    if (Data == Master) {

      Serial.print("\n");
      Serial.print("Password Correct !!");
      Serial.print("\n");
      flag = true;

      lock_state = false;  //Unlocked

      // Turn on relay for 5 seconds
      // digitalWrite(gateValveOutput, LOW);

      // Correct Password
      lcd.print("Pin Matched");
      delay(2000);

      enterCheckMethod();

      //delay(10000);
      //digitalWrite(gateValveOutput, HIGH);
    } else {
      Serial.print("\n");
      Serial.print("Password Incorrect !!");
      Serial.print("\n");
      flag = true;

      lock_state = true;

      // Incorrect Password
      lcd.print("Incorrect Pin");
      delay(2000);
    }

    // Clear data and LCD display
    lcd.clear();
    clearData();
  }
}

void clearData() {
  //Reset data_count
  data_count = 0;
  //Reset Data
  Data = "";
}
```

```
199
200
201 int getLevel() {
202
203   val1 = analogRead(water1);
204   val2 = analogRead(water2);
205   val3 = analogRead(water3);
206   total = val1 + val2 + val3;
207
208   if (total <= 200) {
209     level = 0;
210   } else if (total <= 600) {
211     level = 1;
212   } else if (total <= 700) {
213     level = 2;
214   } else if (total <= 900) {
215     level = 3;
216   } else if (total <= 1300) {
217     level = 4;
218   } else if (total <= 1400) {
219     level = 5;
220   } else if (total <= 1600) {
221     level = 6;
222   } else if (total <= 2000) {
223     level = 7;
224   } else if (total <= 2100) {
225     level = 8;
226   } else {
227     level = 9;
228   }
229
230   Serial.print("water level is: ");
231
232
233   Serial.println(level);
234
235   return level;
236 }
237 void enterThreeDigitNumber() {
238   // Input from keypad for 3 characters
239   lcd.clear();
240   lcd.print("Enter 3 Characters:");
241
242   String keypadInput;       // Array to store the input
243   int inputCharacters = 0;  // Counter for number of characters entered
244
245   while (inputCharacters < 3) {
246     char key = customKeypad.getKey();
247     if (key != NO_KEY) {
248       // Add the key to the input array
249       keypadInput += key;
250       inputCharacters++;
251       // Display the pressed key
252       lcd.clear();  // Clear the display before printing the key
253       lcd.print("Enter 3 Characters:");
254       lcd.setCursor(0, 1);  // Move cursor to the second line
255       lcd.print(keypadInput);
256       delay(200);  // Delay to allow for comfortable keypress rate
257     }
258   }
259
260   // Null-terminate the string
261   keypadInput[3] = '\0';
262
263   // Clear remaining keys from the keypad buffer
264   customKeypad.getKeys();  // Discard any remaining keys in buffer
```

```
265
266   // Display the entered 3-character input
267   lcd.clear();
268   lcd.setCursor(0, 1);
269   lcd.print("Entered: ");
270   lcd.print(keypadInput);
271   delay(10000);
272   // Now you have the 3-character input in keypadInput
273 }
274
275 String takeInput(String d1, String d2, int c) {
276   // Input from keypad for 3 characters
277   lcd.clear();
278   lcd.setCursor(0, 0);
279   lcd.print(d1);
280   lcd.setCursor(0, 1);
281   lcd.print(d2);
282   delay(2000);
283   lcd.clear();
284   lcd.setCursor(0, 1);
285   lcd.print("#: ");
286   String keypadInput = "";  // Array to store the input
287   int inputCharacters = 0;  // Counter for number of characters entered
288
289   while (inputCharacters < c) {
290     char key = customKeypad.getKey();
291     if (key != NO_KEY) {
292       // Add the key to the input array
293       keypadInput += key;
294       inputCharacters++;
295       // Display the pressed key
296       lcd.clear();          // Clear the display before printing the key
297       lcd.setCursor(0, 0);  // Move cursor to the second line
298       lcd.print(keypadInput);
299       delay(200);  // Delay to allow for comfortable keypress rate
300     }
301   }
302
303   // // Null-terminate the string
304   // keypadInput[c] = '\0';
305
306   // // Clear remaining keys from the keypad buffer
307   // customKeypad.getKeys();  // Discard any remaining keys in buffer
308
309   // Display the entered 3-character input
310   lcd.clear();
311   lcd.setCursor(0, 0);
312   lcd.print("Entered: ");
313   lcd.print(keypadInput);
314   delay(3000);
315   return keypadInput;
316 }
317
318
319 void enterCheckMethod() {
320   String keypadInput;
321
322   keypadInput = takeInput("Enter Check", "Characters:", 1);
323
324   if (keypadInput == "A") {
325     gotoManualCheck();
326   } else if (keypadInput == "B") {
327     gotoDynamicCheck();
328   } else if (keypadInput == "C") {
329
330   } else {
```

```
    return;
  }

  // Now you have the 3-character input in keypadInput
}


void gotoManualCheck() {

  int getWaterL = atoi(takeInput("Enter Amount Of", "Water:", 3).c_str());
  int getTractionTime = atoi(takeInput("Enter Traction", "Time:", 3).c_str());
  int getGateValveOpenDuration = atoi(takeInput("Enter Gate Valve", "Open Duration:", 2).c_str());

  int pumpSpeed = 6;  // 1 min -> 6 litre

  String getWaterLStr = String(getWaterL);
  String getTractionTimeStr = String(getTractionTime);
  String getGateValveOpenDurationStr = String(getGateValveOpenDuration);

  lcd.clear();
  // lcd.print(getWaterLStr);
  // lcd.print(" ");
  // lcd.print(getTractionTimeStr);
  // lcd.print(" ");
  // lcd.print(getGateValveOpenDurationStr);
  // delay(2000);

  int pumpTime = getWaterL * 60 / pumpSpeed;  //Calculated Time in Seconds
  Serial.print(pumpTime);
  delay(500);
  turnOnPump(pumpTime);  //Takes input in second

  runTraction(getTractionTime * 60);  //Run traction

  turnOnGateValve(getGateValveOpenDuration * 60);  //Takes input in second

  String again = takeInput("Do you want to ", "check again?", 1);

  if (again == "1") {
    enterCheckMethod();
  } else {
    return;
  }
}


void runTraction(int getTractionTime) {

  for (int i = getTractionTime; i > 0; i--) {
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Traction Running");
    lcd.setCursor(0, 1);
    lcd.print("T:");
    lcd.print(i);
    lcd.print("s");
    delay(1000);  // Wait for one second
  }


  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Traction Finshed");
  delay(2000);
  lcd.clear();
}
```

```
397
398 void turnOnPump(int pumpTime) {
399   digitalWrite(pumpMotorOutput, HIGH);
400
401   for (int i = pumpTime; i > 0; i--) {
402     lcd.clear();
403     lcd.setCursor(0, 0);
404     lcd.print("Pump running");
405     lcd.setCursor(0, 1);
406     lcd.print("T:");
407     lcd.print(i);
408     lcd.print("s");
409     delay(1000);  // Wait for one second
410   }
411
412   digitalWrite(pumpMotorOutput, LOW);
413
414   lcd.clear();
415   lcd.setCursor(0, 0);
416   lcd.print("Pump stopped");
417   delay(2000);
418   lcd.clear();
419 }
420
421
422 void turnOnGateValve(int getGateValveOpenDuration) {
423   digitalWrite(gateValveOutput, LOW);
424
425   for (int i = getGateValveOpenDuration; i > 0; i--) {
426     lcd.clear();
427     lcd.setCursor(0, 0);
428     lcd.print("Draining");
429     lcd.setCursor(0, 1);
430     lcd.print("T:");
431     lcd.print(i);
432     lcd.print("s");
433     delay(1000);  // Wait for one second
434   }
435
436   digitalWrite(gateValveOutput, HIGH);
437
438   lcd.clear();
439   lcd.setCursor(0, 0);
440   lcd.print("Draining stopped");
441   delay(2000);
442   lcd.clear();
443 }
444
445
446 int calculatePumpTime(int getWeight, int perHeight) {
447   double waterL = getWeight/10 ;        //in gm
448   double pumpHeight = waterL * perHeight;  //in mm
449   Serial.println("PumpHeight:");
450   Serial.println(pumpHeight);
451   int iterations = 5;
452   int dist = (sonar.ping_median(iterations) / 2) * 0.343;
453   Serial.println(dist);
454   digitalWrite(pumpMotorOutput, HIGH);
455   int i=1;
456   while ( (MAX_DISTANCE - dist) <= pumpHeight) {
457     Serial.println(dist);
458     lcd.clear();
459     lcd.setCursor(0, 0);
460     lcd.print("Pump running");
461     lcd.setCursor(0, 1);
462     lcd.print("T:");
```

```
    lcd.print(i);
    lcd.print("s");
    delay(1000);  // Wait for one second
    dist = (sonar.ping_median(iterations) / 2) * 0.343;
    i++;
  }

  digitalWrite(pumpMotorOutput, LOW);

  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Pump stopped");
  delay(2000);
  lcd.clear();
}


void turnOnGateValveDynamic() {
  digitalWrite(gateValveOutput, LOW);
  int iterations = 5;
  int dist = (sonar.ping_median(iterations) / 2) * 0.343;
  int i=1;
  while (dist<=175) {
    Serial.println(dist);
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Drain running");
    lcd.setCursor(0, 1);
    lcd.print("T:");
    lcd.print(i);
    lcd.print("s");
    delay(1000);  // Wait for one second
    dist = (sonar.ping_median(iterations) / 2) * 0.343;
    i++;
  }

  digitalWrite(gateValveOutput, HIGH);

  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Draining stopped");
  delay(2000);
  lcd.clear();
}


void gotoDynamicCheck() {

  int getWeight = atoi(takeInput("Enter Patient", "Weight(KG):", 3).c_str());
  int getTractionTime = atoi(takeInput("Enter Traction", "Time:", 3).c_str());
  // int getGateValveOpenDuration = atoi(takeInput("Enter Gate Valve", "Open Duration:", 2).c_str());

  int perHeight = 25; //actualy 12.75( 127.5 for 5kg)  // x mm per 1 kgram

  String getWeightStr = String(getWeight);
  String getTractionTimeStr = String(getTractionTime);
  // String getGateValveOpenDurationStr = String(getGateValveOpenDuration);

  lcd.clear();
  // lcd.print(getWaterLStr);
  // lcd.print(" ");
  // lcd.print(getTractionTimeStr);
  // lcd.print(" ");
  // lcd.print(getGateValveOpenDurationStr);
  // delay(2000);

```

```
529    calculatePumpTime(getWeight, perHeight);
530
531    runTraction(getTractionTime * 60);  //Run traction
532
533    turnOnGateValveDynamic();  //dynamic
534    String again = takeInput("Do you want to ", "check again?", 1);
535
536    if (again == "1") {
537      enterCheckMethod();
538    } else {
539      return;
540    }
541  }
```