**HydroChill – Smart Cooling System**

Description:

HydroChill is a compact, portable cooling system that uses a **Peltier module (TEC1-12706)** for cooling, controlled by an Arduino Nano. The project integrates temperature sensing, fan control, and safety features to maintain effective cooling. It is designed as a low-power, DIY personal cooling solution.

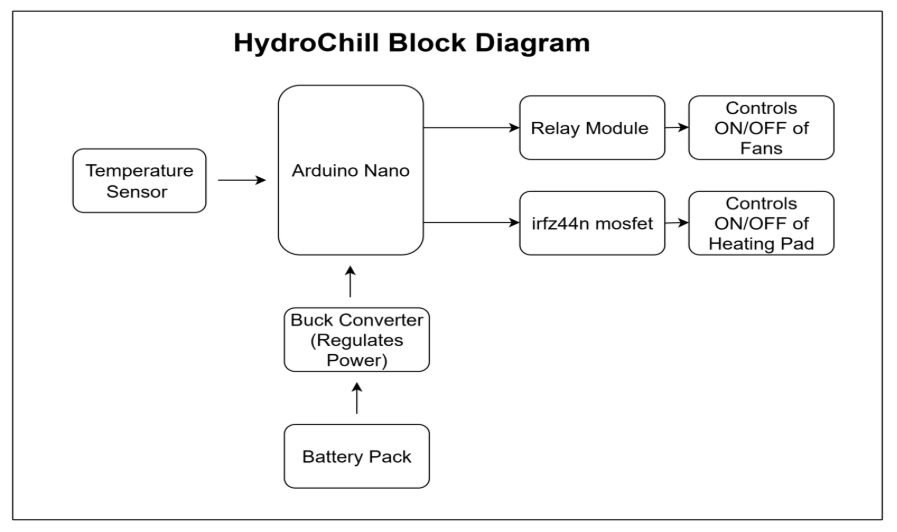
**Materials Used:**

|  |  |  |
| --- | --- | --- |
| **S. No** | **Components** | **Description** |
| 1 | Arduino Nano | A microcontroller that controls all the functions of the system. |
| 2 | DS18B20 Temperature Sensor | Sensor for temperature detection. |
| 3 | Carbon Heating Pads | It Generate warmth in cold conditions. |
| 4 | Mini Fans (5V or 12V) | Fans for airflow and cooling. |
| 5 | IRFZ44N | MOSFET for power control. |
| 6 | Li-po Battery | Rechargeable battery for power. |
| 7 | Thermoelectric Peltier Refrigeration Cooling System DIY Kit | Peltier module for cooling. |
| 8 | Jacket | Wearable jacket to house the system. |
| 9 | Heat Sink & Blower Fan | Dissipates heat from the hot side |
| 10 | Step-Down Buck Converter (LM2596) | Regulates voltage |
| 11 | Switches | Manual ON/OFF control |

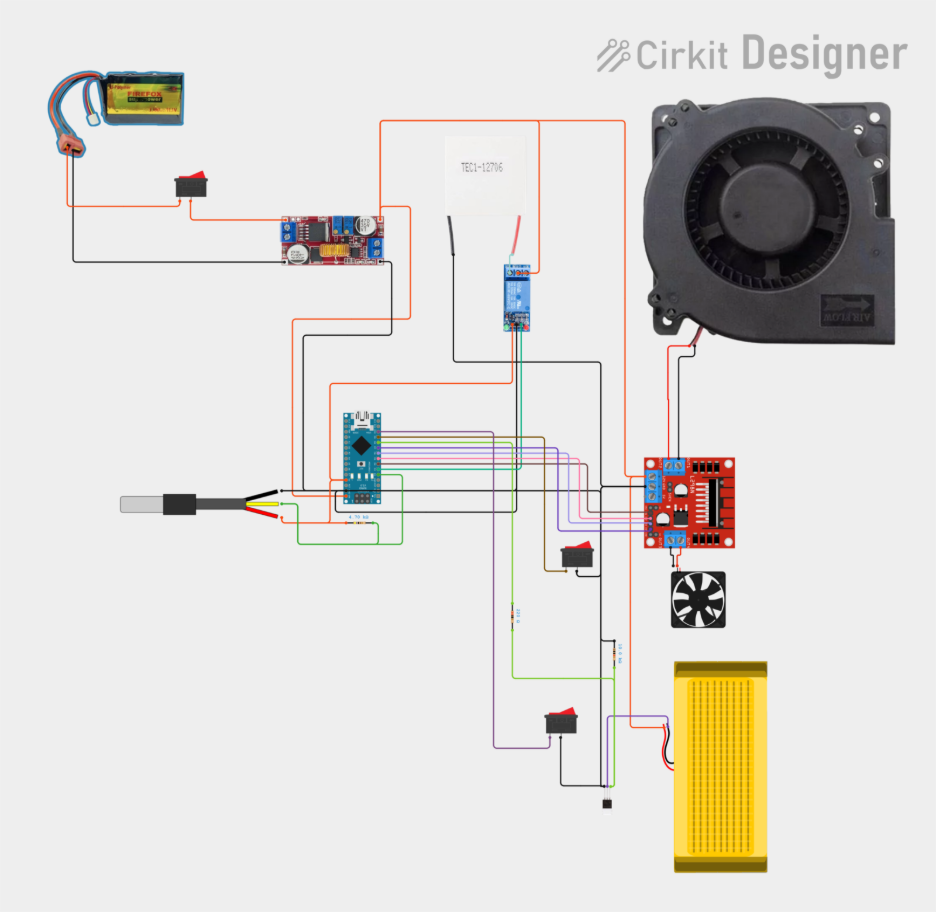
**How to Explain the Project:**

* In hot weather, people look for a small and portable cooling solution.
* Traditional air conditioners are bulky, heavy, and consume a lot of power.
* HydroChill is designed as a compact cooling system that runs on a small battery.
* It uses a **Peltier module**:
* When electricity passes through, one side becomes **cold** and the other side becomes **hot**.
* A fan and heat sink are attached to the hot side so that the cold side can be used effectively for cooling.
* The system is controlled by an **Arduino Nano** which acts as the brain.
* A **temperature sensor** is used to monitor the heat level.
* When the sensor detects that the temperature is above the set limit, Arduino switches ON the Peltier module and fans.
* When the temperature comes back down, Arduino switches OFF the Peltier and fans to save battery.
* Power is supplied through a **battery pack**, and a **buck converter** ensures stable voltage.
* **Relays and switches** are used to safely control the Peltier module and fans.

**Block Diagram:**

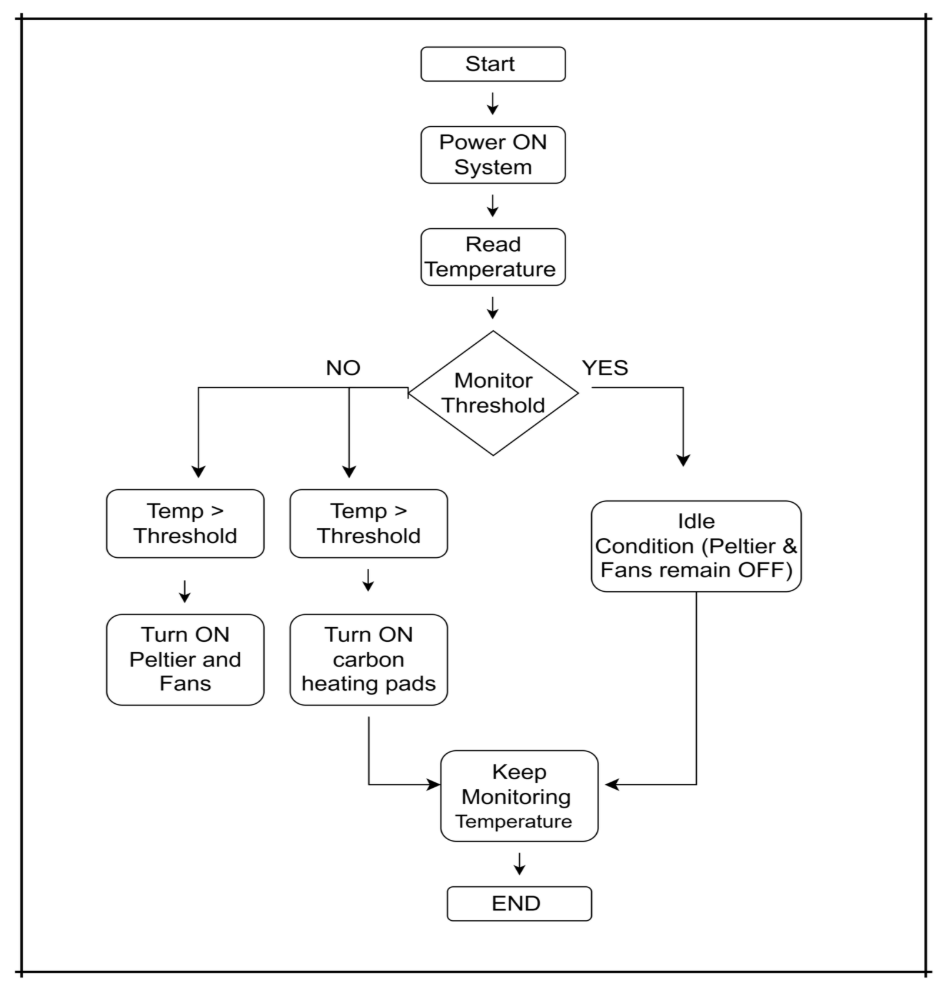


**Circuit Diagram:**



[Circuit Diagram](https://app.cirkitdesigner.com/project/def4286f-e02e-4141-aeea-7321947f72d8)

**Flow Chart:**



**Code:**

#include <Wire.h>

#include <LiquidCrystal\_I2C.h>

#include <OneWire.h>

#include <DallasTemperature.h>

// Pin definitions

#define ONE\_WIRE\_BUS 3    // DS18B20 data pin

#define RELAY\_PIN 2       // Relay control pin (LED via relay)

#define L298\_IN1 11       // L298N IN1 Fan A

#define L298\_IN2 10       // L298N IN2 Fan A

#define L298\_IN3 9        // L298N IN3 Fan B

#define L298\_IN4 6        // L298N IN4 Fan B

#define BUTTON\_COOL\_PIN 7 // Cooling override button

#define BUTTON\_HEAT\_PIN 4 // Heating override button

#define HEAT\_PIN 8        // Heating pad MOSFET gate

// DS18B20 setup

OneWire oneWire(ONE\_WIRE\_BUS);

DallasTemperature sensors(&oneWire);

// LCD setup

LiquidCrystal\_I2C lcd(0x27, 16, 2);

void setup() {

  pinMode(RELAY\_PIN, OUTPUT);

  pinMode(L298\_IN1, OUTPUT);

  pinMode(L298\_IN2, OUTPUT);

  pinMode(L298\_IN3, OUTPUT);

  pinMode(L298\_IN4, OUTPUT);

  pinMode(HEAT\_PIN, OUTPUT);

  pinMode(BUTTON\_COOL\_PIN, INPUT\_PULLUP);

  pinMode(BUTTON\_HEAT\_PIN, INPUT\_PULLUP);

  // Start with everything OFF

  digitalWrite(RELAY\_PIN, HIGH);

  digitalWrite(L298\_IN1, LOW);

  digitalWrite(L298\_IN2, LOW);

  digitalWrite(L298\_IN3, LOW);

  digitalWrite(L298\_IN4, LOW);

  digitalWrite(HEAT\_PIN, LOW);

  Serial.begin(9600);

  sensors.begin();

  lcd.init();

  lcd.backlight();

  lcd.clear();

  lcd.setCursor(0, 0);

  lcd.print("Temp Controller");

  delay(2000);

  lcd.clear();

}

// --- Control functions ---

void controlFans(bool enable) {

  if (enable) {

    digitalWrite(L298\_IN1, LOW);

    digitalWrite(L298\_IN2, HIGH);

    digitalWrite(L298\_IN3, LOW);

    digitalWrite(L298\_IN4, HIGH);

    Serial.println("Fans ON");

  } else {

    digitalWrite(L298\_IN1, LOW);

    digitalWrite(L298\_IN2, LOW);

    digitalWrite(L298\_IN3, LOW);

    digitalWrite(L298\_IN4, LOW);

    Serial.println("Fans OFF");

  }

}

void controlHeater(bool enable) {

  digitalWrite(HEAT\_PIN, enable ? HIGH : LOW);

  Serial.print("Heater: ");

  Serial.println(enable ? "ON" : "OFF");

}

void controlRelay(bool enable) {

  digitalWrite(RELAY\_PIN, enable ? LOW : HIGH); // Active LOW relay

  Serial.print("LED: ");

  Serial.println(enable ? "ON" : "OFF");

}

// --- Update LCD ---

void updateLCD(float tempC, String mode) {

  lcd.clear();

  lcd.setCursor(0, 0);

  lcd.print("Mode: " + mode);

  lcd.setCursor(0, 1);

  if (tempC == DEVICE\_DISCONNECTED\_C) {

    lcd.print("Temp: ERR");

  } else {

    lcd.print("Temp: ");

    lcd.print(tempC, 1);  // 1 decimal place

    lcd.write(223);       // degree symbol

    lcd.print("C");

  }

}

// --- Main loop ---

void loop() {

  bool manualCool = (digitalRead(BUTTON\_COOL\_PIN) == LOW);

  bool manualHeat = (digitalRead(BUTTON\_HEAT\_PIN) == LOW);

  sensors.requestTemperatures();

  float tempC = sensors.getTempCByIndex(0);

  // If both switches are pressed → LOCK state

  if (manualCool && manualHeat) {

    controlFans(false);

    controlHeater(false);

    controlRelay(false);

    updateLCD(tempC, "LOCK");

    Serial.println("Both switches pressed → LOCK");

  }

  else if (manualCool) {

    // Force Fans ON, Heater OFF

    controlFans(true);

    controlHeater(false);

    controlRelay(true);

    updateLCD(tempC, "MANUAL COOL");

  }

  else if (manualHeat) {

    // Force Heater ON, Fans OFF

    controlFans(false);

    controlHeater(true);

    controlRelay(false);

    updateLCD(tempC, "MANUAL HEAT");

  }

  else {

    // Automatic mode

    if (tempC == DEVICE\_DISCONNECTED\_C) {

      controlFans(false);

      controlHeater(false);

      controlRelay(false);

      updateLCD(tempC, "ERROR");

    }

    else if (tempC > 30.0) {

      controlFans(true);

      controlHeater(false);

      controlRelay(true);

      updateLCD(tempC, "AUTO COOL");

    }

    else if (tempC < 24.0) {

      controlFans(false);

      controlHeater(true);

      controlRelay(false);

      updateLCD(tempC, "AUTO HEAT");

    }

    else {

      controlFans(false);

      controlHeater(false);

      controlRelay(false);

      updateLCD(tempC, "IDLE");

    }

  }

  delay(500);

}

**Future Scope:**

* **Smart App Control:** Integrate Bluetooth or Wi-Fi to monitor and control the jacket temperature in real-time using a smartphone app.
* **Advanced Temperature Management:** Use multiple sensors for more accurate detection of body and ambient temperature for smarter heating and cooling.
* **Adaptive AI Comfort:** Implement AI algorithms that learn user preferences and automatically adjust temperature settings.
* **Improved Power Efficiency:** Optimize power usage of heating pads, fans, and Peltier modules to extend battery life.
* **Lightweight & Flexible Design:** Use flexible Peltier modules and lighter materials to improve comfort and wearability.