# **Dawson College**

# **Electrical Engineering Technology Department**

# **Introduction to Internet of Things**

## **Project Name:**

## Basic Linear Supply Management with Arduino UNO R3

## **Team Members:**

## Victor Matta-Myers Student (ID: 2437241), Evan Matulina (ID: 1732620)

## May 13, 2025

## 2. Project Description

For most who like to work on electronics engineering projects in their free time, a home lab is a necessity. However, a home lab can get to be pricey, luckily the components to make them aren’t. So, a lot of people resort to DIY projects including us.

And if you are looking for a cheaper alternative to a power supply you are looking in the right place, by using an Arduino, LCD display and a heat sensor you can monitor the voltage output of your own DIY variable power supply.

It displays Voltage output and temperature to an LCD display for options to include custom messages and warnings in the near future.

**How it works:**

* Using large value resistors, a voltage divider (to not overload the Arduino) can read the voltage output of your power supply. It divides the incoming voltage from the power supply by 10 to read up to 50V without damaging the Arduino.
* Using an 18B20 temperature sensor it monitors heat dissipation of the voltage regulators inside of your power supply. It then uses that data to control fans to help maintain a tight range.
* Using a LCD display it allows the user to see the voltage output and temperature inside the unit

**Final assembly diagram:**

Use any drawing tool of your preference. This is meant to have students think about practical implications for their design. The drawing below was done on PowerPoint. Use the diagram to help with your project description.

Figure 1 - Project diagram.figure above indicates a general block diagram

## 3. Circuit Diagram

**Inputs:**

* Temperature Sensor(DS18B20)
* Custom voltage divider network

**Outputs:**

* LCD Display
* Fans

Below is a description of the hardware connections, which are also shown graphically using Fritzing (or a similar CAD tool). Note how schematic is **neat, labelled and colour-coded,** so the instructor can follow the circuit at a glance.

Table 1 - Circuit Connections

|  |  |  |
| --- | --- | --- |
| **component** | **Pinout for named component** | Notes |
| LCD Display | 1k resistor to ground → VD,  Pin 7 → RS, Pin 8 → RW, K→GND Pin 9 → D4,  Pin 10 → D5, Pin 11 → D6,  Pin 12 → D7, A→VCC | Standard connections to display voltage output and temperature inside component |
|  |  |  |
| Temp Sensor | GND→Pin 1, VCC→Pin 2,  Vout → Pin 3 | Temperature sensor to monitor heat in unit |
| Volt Sensor | Voltage Divider node A → A5 | Using resistors to lower voltage entering Arduino board used for detecting voltage output |
| Fan Control | Pin 5 in PWM mode | This is used to modulate the GATE of a TIP110 Darlington transistor pair. Since the Arduino uno R3 is limited to 40mA output it needs a transistor for switching big loads. |

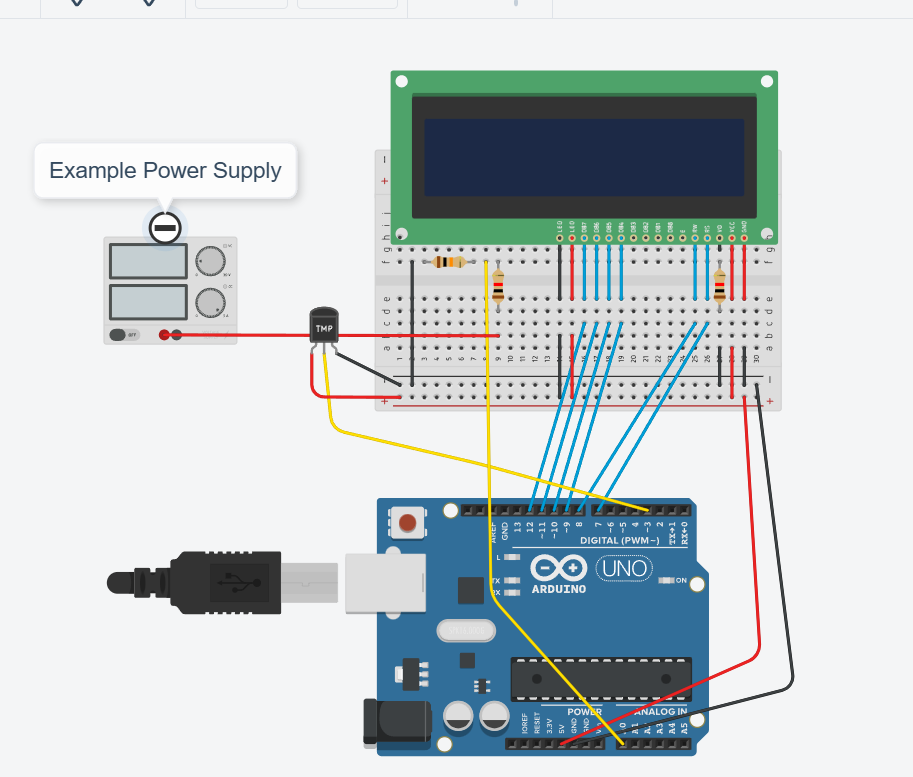


Figure 2 - Circuit Connections Diagram. Image from Fritzing, educational purpose only.

**Notes about the circuit assembly:**

* 1. Resistors can be any value as long as the max voltage does not exceed the maximum voltage the Arduino uno board can handle.
  2. The diagram is merely to give a rough idea on how to accomplish the setup. Some tinkering is required depending on the type of supply used.

## 4. Code Documentation

**4.1 Library Used**

#include <LiquidCrystal.h> handles the LCD display and its functions

#include <OneWire.h> Handles the temperature sensor

#include <DallasTemperature.h> simplifies onewire library

**4.2 Global Constants & Pins**

Below is the list of constant variables used in the code. As constants they do not change during the running of the code, and as global they can be used inside and outside functions.

Table 2 - Global constants and Pins.

|  |  |  |
| --- | --- | --- |
| **Constant / Pin** | **Purpose** | **Typical Value** |
| *LCD Display pins* | Just various pins to power and send information to the display | 1k resistor to ground → VD,  Pin 7 → RS, Pin 8 → RW,  K→GND Pin 9 → D4,  Pin 10 → D5, Pin 11 → D6,  Pin 12 → D7, A→VCC |
| *Temp Sensor* | Measures temperature and sends it to the arduino board | Vout → Pin 3 |

**4.3 Function Descriptions and Responsibilities**

Table 3 - Function descriptions and Responsibilities.

|  |  |
| --- | --- |
| **Functions** | **Descriptions & Responsibilities** |
| float Temper() | • measures temperature  • using the measured temperature it determines needed fan state and returns that value to be used later • also returns temperature to be used later   * Controls the fans as well |
| float Voltage() | • measures voltage • maps the voltage to accommodate for Arduinos max voltage input • returns this voltage value for later use |
| void display(float displayVolt, float temp) | • displays the before mentioned values that being temperature and voltage for later use using prefixes to make the values easy to read and understand for the user |
| void loop() | • Calls the three functions • Clears The LCD display before updates for clearer display |

**4.4 Code Explanation**

While the code can be found on our [GitHub](https://github.com/vishaywesay/shiny-potato) , an explanation of how the code works is shown below.

1. **Temperature sensor** – measures temperature and uses the measured temperature to determine the needed fan state and returns that value to be used later.
2. **Fan** – Cools the unit to ensure all safe conditions are met and varies depending on the temperature inside to meet ideal conditions.
3. **Voltage** – *the code measures voltage from a voltage divider and translates it so that the Arduino can handle and display it to be used later on. This value is used later on in the code.*
4. **LCD Display** – uses the LCD display to display the temperature and voltage output to be readable to the user and accessible to the user

Together these steps create an autonomous monitor: displays the voltage output and in the unit. ￼

/\*

Dawson College

Electronics Engineering Technology

Lab Exam

Authors: Evan Matulina and Victor Matta-Myers

Project: Arduino Managment System for a Linear Power Supply

\*/

//All the libraries needed for this code to run

#include <DallasTemperature.h>

#include <LiquidCrystal.h>

#include <OneWire.h>

//Assigning of the variables and constants as well as one object created

const int rs = 7, en = 8, d4 = 9, d5 = 10, d6 = 11, d7 = 12;

LiquidCrystal lcd(rs, en, d4, d5, d6, d7); // Create an LCD object

int VoltPin = A5; //used for analog in

int fanPin = 5; //PWM control

float volt; //volt will be a variable that requires some precision. Float is required

float temp; //can be changed to int if memory is an issue

int fan;

int x;

int y;

int z;

const int Hspeed = 255; //This is used to set pin controlling fan to full on

const int Mspeed = 200; // Value for setting fan to half speed on analog pin control

const int Lspeed = 102; // Value for min fan speed on analog pin control pin for TIP110

#define tempPin 3

OneWire oneWire(tempPin); // Initialize OneWire protocol on pin tempPin.

DallasTemperature sensors(&oneWire); // DallasTemperature library to simplifies

void setup() {

//Starting the serial connections and defining pins

Serial.begin(9600);

pinMode(VoltPin, INPUT);

sensors.begin(); //Initialize the temp sensor

pinMode(tempPin, INPUT);

pinMode(fanPin, OUTPUT);

lcd.begin(16, 2); // Initialize LCD type (x columns, y rows)

}

float Voltage() {

int VoltValue = analogRead(VoltPin);

analogRead(VoltPin);

//Analog read from the voltage divider using A to D conversion

float volt = VoltValue \* (48.7 / 1023.0); //The number divided by 1023.0 is the calibration constant

//Mapping the voltage

return volt;

}

//This function is for the temperature readout

float Temper() {

analogRead(tempPin); //Reads the pin used for temp

sensors.requestTemperatures(); // Function requesting Temperature

temp = sensors.getTempCByIndex(0); //Call a fonction from the DallasTemp lib

int fanSpeed = 0; //Initial value of fanSpeed is zero as a placeholder. Replaed with speed value

//This is the logic for the fan control

if (temp > 30) {

fanSpeed = Hspeed; //Fans at full speed

} else if (temp > 25) {

fanSpeed = Mspeed; //Half speed when greater than 25 degrees C

} else {

fanSpeed = Lspeed; //Puts the fans to a minimum speed

}

analogWrite(fanPin, fanSpeed); //This is where the value is sent out to the pin

return temp; //For use later

}

void Serialdisplay(float volt, float temp) {

lcd.clear(); // Clear the LCD before displaying new values

lcd.print("Temperature: ");

lcd.print(temp); //Pulls the vlaue

lcd.print(" °C ");

lcd.setCursor(0, 1); // Move to the second line

lcd.print("Voltage: ");

lcd.print(volt); //Pulls the value

lcd.print(" V ");

}

void loop() {

float currentVoltage = Voltage(); // Get the current voltage

float currentTemperature = Temper(); // Get the current temperature

Serial.print("Voltage: ");

Serial.println(currentVoltage); // Print voltage to Serial Monitor

Serial.print("Temperature: ");

Serial.println(currentTemperature); // Print temperature to Serial Monitor

Serialdisplay(currentVoltage, currentTemperature); // Display on LCD

delay(500); // Delay for half a second

}

## 5. Ethics, Privacy, or Security Disclaimer

Our tool is an assistance but it's important to understand the contents before using it for complete safety. Do not attempt to use, disassemble, modify or open a power supply without knowing the risks involved. This project was developed with a standard 24V DC power supply which poses very little risk to the user.

No data is collected, nor do we plan on ever collecting data, however, the code repository is found on GitHub. Therefore, users are subject to the data collection practices and policies of GitHub

## 6. References

* Elegoo HC-SR04 documentation: <https://www.elegoo.com/>
* Arduino NewPing Library: <https://github.com/microflo/NewPing>
* Arduino Tutorial: Ultrasonic Sensor HC-SR04 - <https://randomnerdtutorials.com/>
* Fritzing - <https://fritzing.org/>
* TIP110 Datasheet - <https://www.onsemi.com/pdf/datasheet/tip110-d.pdf>
* LM317 Datasheet - <https://www.ti.com/lit/ds/symlink/lm317.pdf?ts=1747030104900&ref_url=https%253A%252F%252Fwww.ti.com%252Fproduct%252FLM317>