Svarog

Version 1

Apr 4th, 2025

UNB Fredericton

Name: Samuel Woytiuk

Date: Apr 4th, 2025

Introduction

This project is a thermostat-based pool pump controller named Svarog, after the Slavic god of heat and fire. The device was designed for my father to automate the pool's temperature regulation. Previously, he would need to manually turn the pump on and off while monitoring the temperature, which was time-consuming and inefficient.

Svarog aims to maintain a consistent pool temperature range by automatically controlling the pump based on real-time temperature readings. It is intentionally not an IoT device to keep costs low and because internet connectivity adds unnecessary complexity for this specific use case.

Table of Contents

Resources Used

Hardware

- Sparkfun Readboard
- PowerSwitch Tail II
- ds18b20 – Waterproof Temperature Sensor

Software

- Arduino IDE
- Github

Libraries

- OneWire
- DallasTemperature
- LiquidCrystal

Requirements

This project was built using spare hardware provided by my father, which informed several design decisions—most notably, avoiding IoT capabilities. The setup was focused on affordability, ease of use, and power savings.

Algorithm

1. The DS18B20 sensor reads the water temperature in Celsius.
2. The reading is converted to Fahrenheit.
3. If the temperature drops below 80°F, the pump is activated.
4. If the temperature exceeds 82°F, the pump is deactivated.
5. The PowerSwitch Tail II handles AC switching of the pump safely.
6. The control logic updates every 500 milliseconds to reflect current readings.

Please reference figure 1. Algorithm

Implementation

Depending on the size of your ds18b20 (temperature sensor) and the placement of the heater pump will determine the placement of the device. I suggest placing the temperature sensor in the skimmer of the pool to ensure there is minimal chances of it being disturbed. Ensure that the housing of the device separates it from the elements, possible housing designs will be added in later version reports.

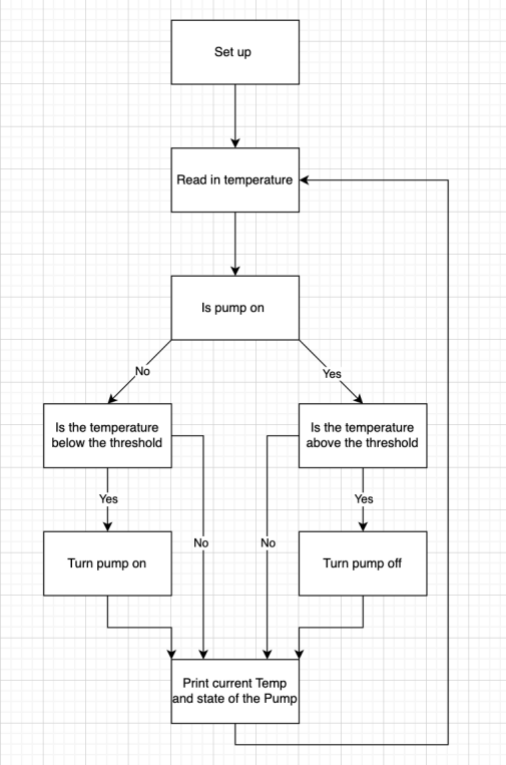

Figure 1 Algorithm

Troubleshooting

The two main issues I had in this project were getting the LCD working and getting the temperature sensor reading in the temperature. Initial difficulties in wiring and initializing the display required reviewing datasheets and community examples to correctly configure the pin layout and use the `LiquidCrystal` library. The DS18B20 sensor would return incorrect values without a 4.7kΩ pull-up resistor and required careful testing with the `OneWire` and `DallasTemperature` libraries to ensure proper utilization.

Leason's Learned

This project provided my with lots of opportunity to put into practice what I had previously learned through the electrical courses earlier in my degree. I learned about the use of more libraries on the Arduino and how easy and clean they make the code. This was the first time working with a bread board on my own and with only the half-sized ones and needing 3 major components connected, I learned quickly that planning out how

each component is going to be connected to the board is a very important step.

Conclusion

Some future work will be adding a user interface, either using the LCD or something on a laptop or phone when the user plugs in their laptop or phone. There will be an additional document update after the 2025 summer with a comparison between the electrical cost from 2024 and 2025 to see how effective this prototype was on the grid to ensure that if there is a noticeable decrease in electricity used and price. As well there will be an updated document for any major changes to the system.

Code

```
#include <LiquidCrystal.h>
#include <OneWire.h>
#include <DallasTemperature.h>

// === LCD Setup ===
LiquidCrystal lcd(13, 12, 11, 10, 9, 8);

// === Optional Temp Sensor Setup ===
#define ONE_WIRE_BUS 7
OneWire oneWire(ONE_WIRE_BUS);
DallasTemperature sensors(&oneWire);
#define TARGET 30
#define UPPER //Upper threshold of the pool temp
#define LOWER //Lower threshold of the pool temp

// === Pump Control Setup ===
#define PUMP_PIN 6

// === Timer Settings ===
const unsigned long pumpOnDuration = 5000;  // 5 seconds ON
const unsigned long pumpOffDuration = 5000; // 5 seconds OFF

// === State Tracking ===
```

```
bool pumpOn = false;
unsigned long lastToggleTime = 0;

void setup() {
  Serial.begin(9600);
  lcd.begin(16, 2);
  sensors.begin();
  pinMode(PUMP_PIN, OUTPUT);
  digitalWrite(PUMP_PIN, LOW);

  lcd.print("Svarog Ready");
  delay(2000);
  lcd.clear();
}

float cToF(float celsius){
  return (celsius * 9/5) + 32
}

void loop() {
  //unsigned long currentMillis = millis();

  /*if (pumpOn && (currentMillis - lastToggleTime >= pumpOnDuration)) {
    pumpOn = false;
    lastToggleTime = currentMillis;
    digitalWrite(PUMP_PIN, LOW);
  } else if (!pumpOn && (currentMillis - lastToggleTime >= pumpOffDuration)) {
    pumpOn = true;
    lastToggleTime = currentMillis;
    digitalWrite(PUMP_PIN, HIGH);
  }*/
  // real temp

  /*
  if (pumpOn && tempC > UPPER){
    pumpOn = false;
    digitalWrite(PUMP_PIN, LOW);
```

```
  } else if (!pumpOn && tempC < LOWER) {
    pumpOn = true;
    digitalWrite(PUMP_PIN, HIGH);
  }
  */
  // === Read in Temperature ===
  sensors.requestTemperatures();
  float tempC = sensors.getTempCByIndex(0); // Shows -127 if no sensor is connected
  float tempF = cToF(tempC);
  // === Pump Control Logic ===
  if (pumpOn && tempC > TARGET){
    pumpOn = false;
    digitalWrite(PUMP_PIN, LOW);
  } else if (!pumpOn && tempC < TARGET) {
    pumpOn = true;
    digitalWrite(PUMP_PIN, HIGH);
  }

  // === LCD Output ===
  lcd.setCursor(0, 0);
  lcd.print("Temp:");
  lcd.print(tempC, 1);
  lcd.print("C  ");
  lcd.print(tempF, 1);
  lcd.print("F ");

  lcd.setCursor(0, 1);
  lcd.print("Pump: ");
  lcd.print(pumpOn ? "ON " : "OFF");

  delay(500); // Update every half second
}
```

References

https://randomnerdtutorials.com/guide-for-ds18b20-temperature-sensor-with-arduino/