

# Heater Control System using Arduino with Timer Interrupt

## 1. Introduction

This project implements a heater control system using Arduino, a DS18B20 temperature sensor, and a relay module to control a simulated heater. In this prototype, an LED is used to represent the heater, providing a visual indication of when the heater is on or off in a real system. It uses a state machine to track and manage system behavior across five states: Idle, Heating, Stabilizing, Target Reached, and Overheat. A TimerOne interrupt triggers periodic temperature sampling every 500 ms, making the system responsive without using blocking delays.

## 2. Sensors and Actuators

### 2.1 Sensors

#### 1. DS18B20 Temperature Sensor:

- **Protocol:** OneWire.
- **Purpose:** Reads temperature for state transitions.

### 2.2 Actuators

- **Relay Module:** Controls heater ON/OFF.
- **LED Indicator:** Provides visual feedback.
- **Buzzer:** Gives an audible signal when the target temperature is reached.

## 3. Communication Protocols

- **One Wire Protocol** - For interfacing the DS18B20 temperature sensor. This single-wire protocol reduces wiring complexity and supports multiple sensors on the same bus.

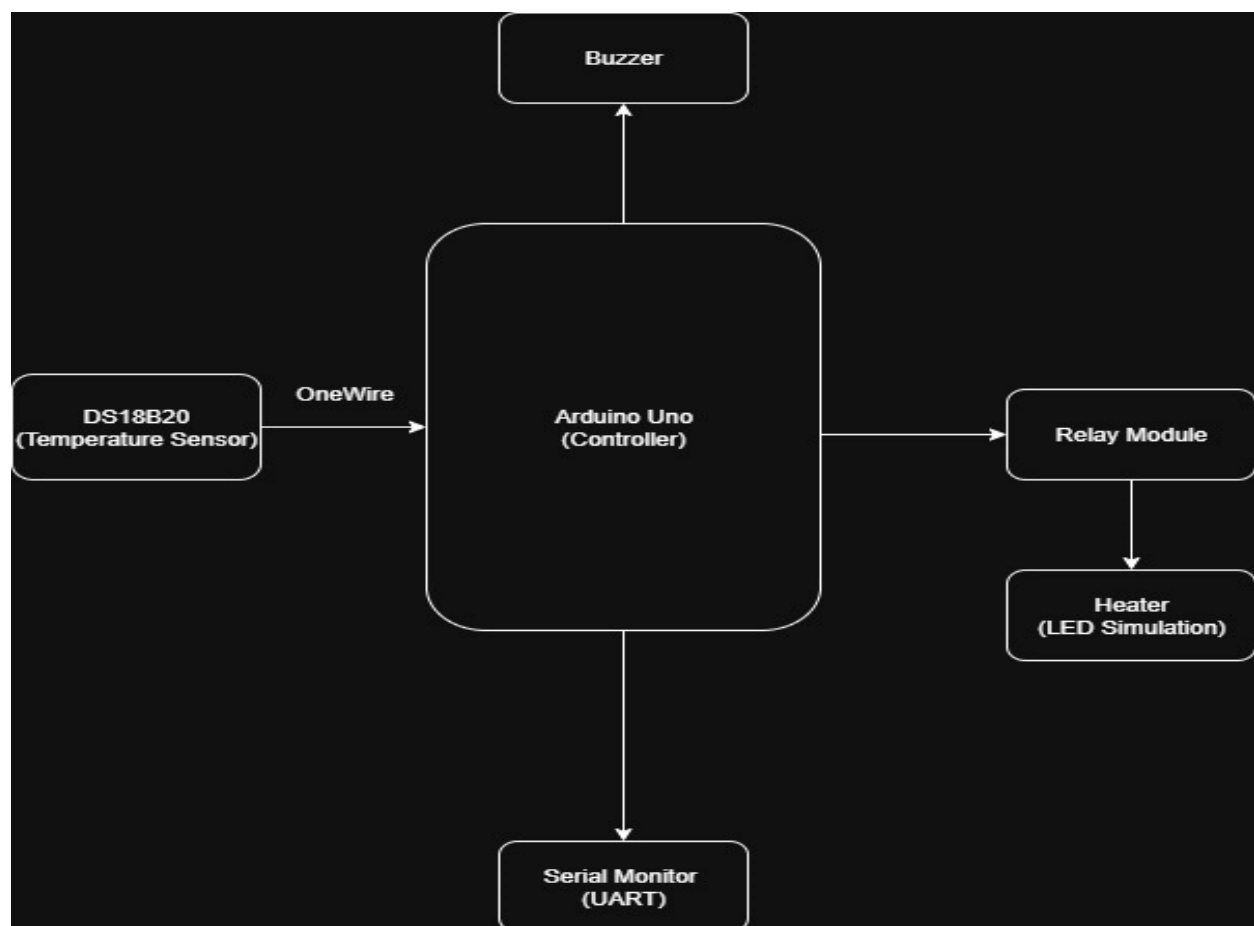
- **UART Protocol** - For logging temperature and state transitions to the Arduino Serial Monitor. This aids in debugging and monitoring.

#### 4. Timer Interrupt

The project uses the TimerOne library to set up a hardware timer interrupt:

- **Interval:** 500 ms
- **Purpose:** Periodically sets a flag (Flag) for non-blocking temperature reading and state updates.
- **Advantage:** Ensures consistent periodic execution without using delay(), improving system responsiveness.

#### 5. Block Diagram



## 6. System States

### State Descriptions

#### 1. Idle State:

- Temperature  $< 30\text{ }^{\circ}\text{C}$ .
- Heater OFF.

#### 2. Heating State:

- Temperature between  $30\text{ }^{\circ}\text{C}$  and  $90\text{ }^{\circ}\text{C}$ .
- Heater ON.

#### 3. Stabilizing State:

- Temperature between  $90\text{ }^{\circ}\text{C}$  and  $98\text{ }^{\circ}\text{C}$ .
- Heater OFF to avoid overshooting.

#### 4. Target Reached State:

- Temperature between  $98\text{ }^{\circ}\text{C}$  and  $100\text{ }^{\circ}\text{C}$ .
- Heater OFF, buzzer beeps periodically.

#### 5. Overheat State:

- Temperature  $> 100\text{ }^{\circ}\text{C}$ .
- Heater OFF for safety.

## 7. Features

- **Non-blocking design:** Uses timer interrupt instead of `delay()`.
- **Real-time logging:** Outputs temperature and system state over Serial.
- **State machine:** Clear modular handling of different temperature ranges.
- **Visual & auditory alerts:** LED & buzzer for state indication.

## 8. Roadmap for Future Enhancements

- **BLE/Wi-Fi + MQTT Integration:** Enable sending temperature and state data to a mobile app or web dashboard using MQTT or REST API for remote monitoring.
- **FreeRTOS Tasks:** Split temperature reading, control logic, and logging into separate tasks for scalability.
- **Multiple Heating Profiles:** Allow dynamic thresholds for different applications.

## 9. Wokwi Simulation

The system was successfully simulated on Wokwi to validate its behavior, including state transitions and relay/buzzer actions.

### Simulation Link:

[View Simulation here](#)

## 10. Conclusion

This project demonstrates a modular, interrupt-driven heater control system using Arduino and DS18B20. The use of a hardware timer enables non-blocking periodic temperature sampling, making the system more responsive and reliable for real-world use.