1. Temperature Sensor

1.1. Sensor Purpose

The DS18B20 temperature sensor is integrated into the ROV to monitor environmental conditions critical to the system's operation. Its primary purpose is to ensure thermal conditions for marine creatures, especially underwater coral reefs. Temperature data is also used to issue alerts if thresholds are exceeded and contribute to long-term performance logging.

1.2. Software Requirements

To interface with the DS18B20 temperature sensor, the following software tools and configurations are required:

- Libraries:
  - OneWire: Manages the 1-Wire communication protocol for data transfer.
  - Dallas Temperature: Provides high-level functions for working with the DS18B20.
- Microcontroller Resources:
  - One GPIO pin for communication.
  - Digital pin configured for bidirectional communication on the 1-Wire bus.

1.3. Overview

The DS18B20 is a 1-Wire digital temperature sensor designed to provide precise temperature readings with minimal wiring. It features four primary data components:

1. 64-bit Lasered ROM:
   Uniquely identifies the sensor on a shared 1-Wire bus, enabling multiple sensors to operate simultaneously on a single communication line. This ROM allows addressing specific sensors for commands.
2. Temperature Sensor:
   Captures temperature measurements and converts them to digital format with a configurable resolution (9 to 12 bits).
3. Nonvolatile Temperature Alarm Triggers (TH and TL):
   Stores threshold values for temperature alarms. These registers are nonvolatile and persist even after power loss. They can be repurposed as general-purpose memory.
4. Configuration Register:
   Contains resolution settings (affecting conversion time) and other control parameters.

The DS18B20 supports two power modes:

- Parasite Power Mode:
  Utilizes energy stored in an internal capacitor charged during high states of the 1-Wire signal.
- External Power Mode:
  Operates with an external 3V–5.5V supply.

Figure (1) explains the sensor's internal architecture and we will discuss later how to use the sensor simply.

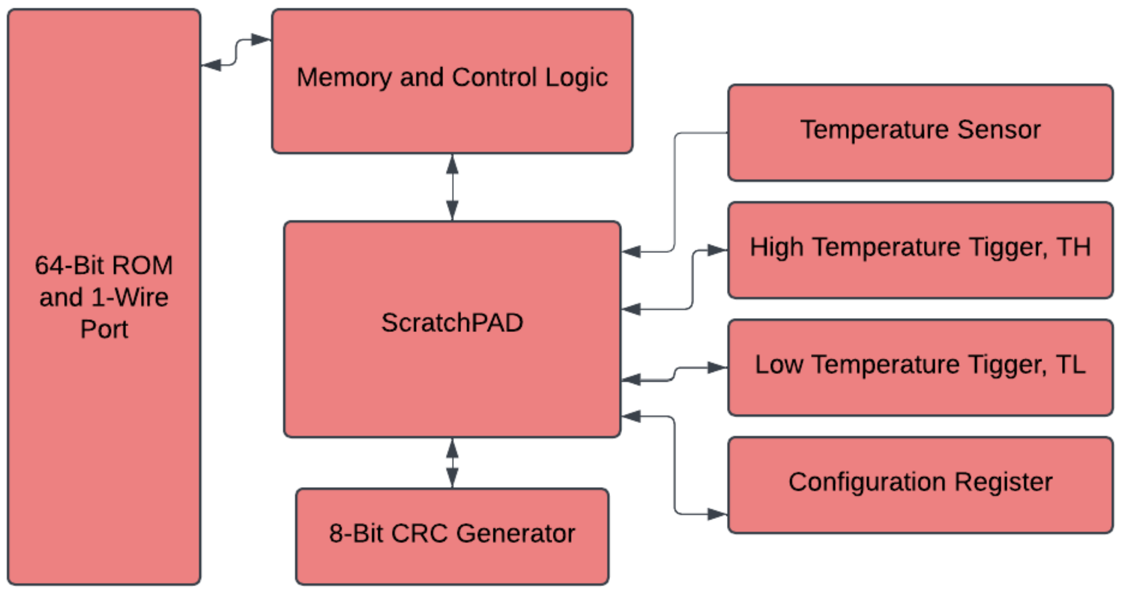

Figure (1)

1.4. Software Architecture

The temperature sensor's role within the software architecture can be divided into the following layers:

1. Data Acquisition Layer:
   - Handles the communication with the DS18B20 sensor using the OneWire protocol.
   - Ensures periodic data requests based on system requirements.
2. Processing Layer:
   - Converts raw temperature readings to human-readable formats (Celsius and Fahrenheit).
   - Applies filtering or averaging for noise reduction.
3. Integration Layer:
   - Sends processed data to subsystems such as the control system (for decision-making) and UI (for display and logging).
4. Error Handling Layer:
   - Detects and addresses disconnection or invalid readings.

1.5. Sensor Interface

1.5.1. Initialization and Configuration

The DS18B20 communicates through a 1-Wire protocol, and the DallasTemperature library simplifies this communication by abstracting the low-level operations and not getting involved in the complicated process mentioned before. We first import the OneWire and DallasTemperature libraries so we can use the functions provided by these libraries and then configure the OneWire GPIO pin connected to the sensor (ex, pin 2) and create an object from both libraries.

Then we initialized the sensor and set the resolution to (9 to 12 bits) higher resolution increases accuracy but requires a longer conversion time. (9 bits: ~93 ms and 12 bits: ~750 ms)

1.5.2. Data Retrieval

Acquiring temperature data involves requesting readings and handling delays for conversion we first start by requesting the temperature data from the sensor as it operates asynchronously, requiring a request before retrieving the data and after that, we fetch the temperature data from the sensor in Celsius or Fahrenheit thanks to DallasTemperature library the provide the functions and attributes to do so

1.5.3. Data Processing and Units Conversion

Raw data from the DS18B20 is converted into human-readable units directly by the DallasTemperature library. Temperature is available in both Celsius and Fahrenheit formats.

Scaling and Calibration are available due to systematic error by adding or subtracting offsets.

1.5.4. Integration

We can use the temperature data in logs for performance analysis and maintenance and send warnings or notifications via the UI or telemetry system in case of overheating

1.5.5. Error Handling

The sensor may be disconnected so we check it and print if it’s connected or make It restart if it’s disconnected. Another error may happen if the temperature data exceeds plausible temperature ranges, flag it for review.

1.6. Pseudocode

1.6.1. Header file

```
DECLARE class TemperatureSensor
    METHOD: begin()
    METHOD: setResolution(resolution)
    METHOD: readTemperature()
ENDCLASS
```

1.6.2. Source file

```
DEFINE TemperatureSensor
    METHOD: begin()
        START OneWire and DallasTemperature objects
    METHOD: setResolution(resolution)
        SET desired resolution for the sensor
    METHOD: readTemperature()
        REQUEST temperature measurement
        RETURN temperature value
ENDDEFINE
```

1.6.3. Main file

```
IMPORT TemperatureSensor library
DECLARE TemperatureSensor object sensor
SETUP:
    CALL sensor.begin()
    CALL sensor.setResolution(12 bits)
LOOP:
    CALL sensor.readTemperature()
    PRINT temperature value
ENDLOOP
```