Critical Assessment Document

IoT Implementation for a Harsh Temperature Monitoring System

University of South Florida
Department of Electrical Engineering
EE Senior Design II – Fall 2022
Dr. Alexandro Castellanos, Ph.D.

December 1, 2022

Team Members

Brandon Collins
Ashley Porter
Paul Polgar
Steve Lambropoulos

Abstract

Many construction companies use large industrial equipment, such as concrete crushers and excavators, on their job sites to accomplish large-scale tasks. However, on-site personnel are unable to monitor the internal temperature of this equipment to know if it is operating outside of the desired temperature ranges. When the internal temperature of the equipment fluctuates out of normal temperature ranges, this could potentially make for a dangerous environment.

The IoT Implementation for Harsh Temperature Monitoring System is an innovative system that will allow companies to maintain and improve the longevity of their operational equipment, such as concrete crushers. Our product will monitor and alert onsite personnel if components on equipment is operating outside of their desired temperature range by using a temperature sensor to monitor the temperature of the equipment, send this data back to a server to be analyzed and provide real time measurements for operators.

Project Objective

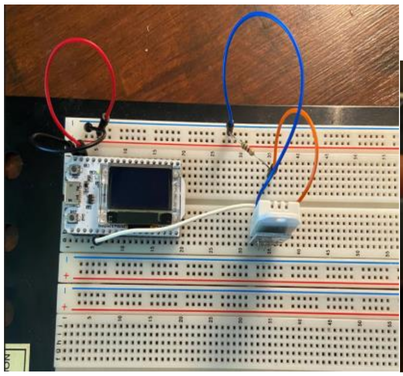
The overall goal in designing the IoT Implementation system was to create an assembly that successfully and efficiently met all functional requirements designed at the beginning of the fabrication process. The device serves to accomplish the following objectives:

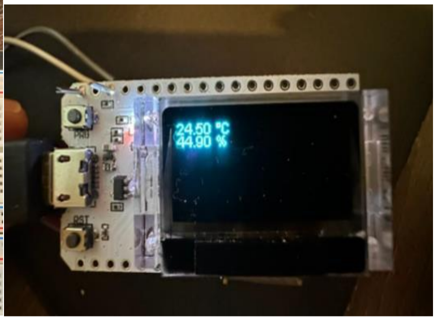
- Device can operate in harsh environments, including being resistant against dust, heat, cold, and vibrations
- Is able to sustain itself with batteries for 6 months minimum without needing to be recharged
- Is configurable using a Bluetooth connection
- Transmits data using a cellular connection to an offsite server
- Maintains the longevity of a company’s large operational equipment by ensuring the equipment is operating at a safe temperature
- Sends alerts back to a server if operational temperature fluctuates out of range
- Alerts onsite personnel of unsafe temperature conditions of operating equipment
- Ultimately serves to eliminate a potentially hazardous situation, while protecting the quality and durability of a company’s equipment

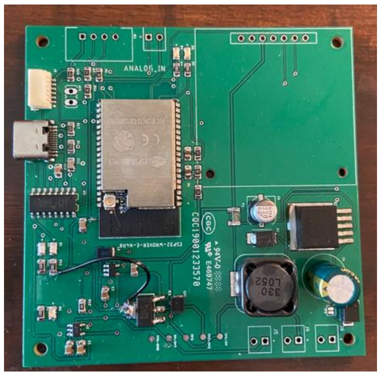
Test Plan

Test Plan #1 – PCB and Temperature Sensor

While the PCB was being designed, we were able to use the ESP32 Heltec microcontroller (another version of the ESP32 used in the PCB design) and a DHT22 thermistor temperature and humidity sensor, which we interfaced on a breadboard and connected via USB-C to our computers to test. To advance from this prototype, on which we tested the firmware and the ability to gather data from the temperature sensor and microcontroller, we developed a detailed PCB design that interfaced all components with the ESP32 microcontroller and a 24VAC, 5V, USB-C, or battery power source. This design allowed all components to be soldered and limited the possibilities of errors that would occur using circuits and a breadboard. We encountered missing grounds and connections with the first PCB design, so another revision was made. The second revision was better, but the final revision yielded no errors and included all necessary components and power sources, while minimizing power draw and allowing for operation in a variety of environments. The thermistor temperature sensor is self-adhesive to the surface of interest and yielded accurate results when tested. The thermistor and ESP32 interface as required.





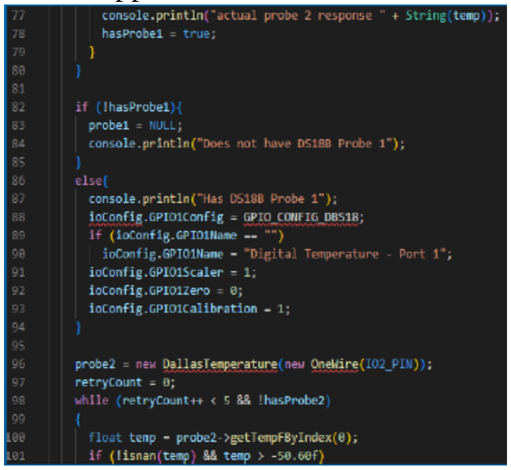


Test Plan #2 – Power Consumption

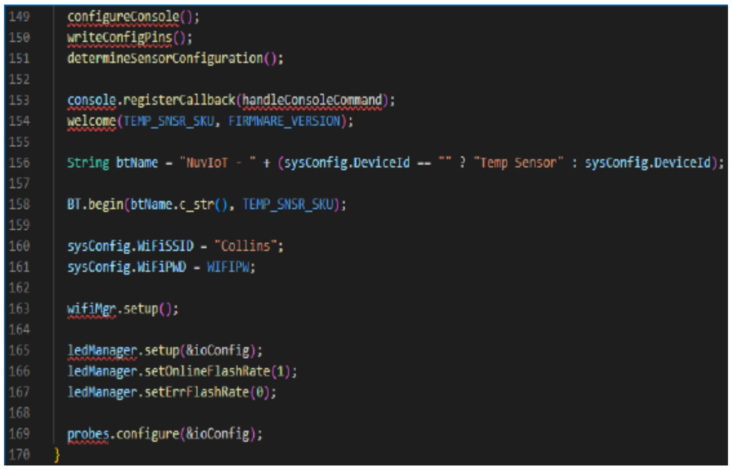
By estimating the power of all modules and analyzing their required voltages, the required current carrying capability of each power supply component was estimated. Voltage regulators were used to easily step down the voltage from 24V to 5V bus, and 5V to 3.3V. When tested with a multimeter, all voltages from power supplies were accurate, with only about a ±0.1V margin of error. In order for the system to operate on either a 24VAC, 5V, USB-C or on battery power, we separated the 5V bus into two sections, split by a Schottky diode. One section connected the 5V to 3.3V regulator and other section connected the Li-po to 5V booster. The circuit knew that if the 5V bus connected to external source was powered, then battery charging operations would be allowed. If the bus was not powered, then the Li-po charger shouldn't operate. This prevented a circular response of the battery attempting to charge itself. Testing of the different power supplies based on the connections were carried out, and each power supply worked. We also implemented a deep sleep method for programmable peripherals ICs using firmware to conserve power.

Test Plan #3 – Firmware

Firmware for the device is built on top of the provided *Software Logistics* libraries that are pre-optimizing and configured for use with the ESP32 microcontroller and peripherals. Firmware has a standard entry point through its initialization sequence that operates on a loop during standard operation. The firmware facilitates interaction between hardware, server, and mobile app. During boot/setup phase of the device the firmware determines the pin configuration of hardware, configures file system, serial debugging interface, cellular modem, then loads currently stored device configuration, determines configuration of temperature sensors, initializes Bluetooth radio, configures, and makes connection to Wi-Fi and/or cellular. The device was tested using the thermistor and heating it to a temperature above the pre-configured acceptable temperature range. When the temperature became out of range, the firmware allowed the thermistor and ESP32 to communicate and send data back to the server. An alert was sent to associated personnel via the mobile app. Firmware was successful.



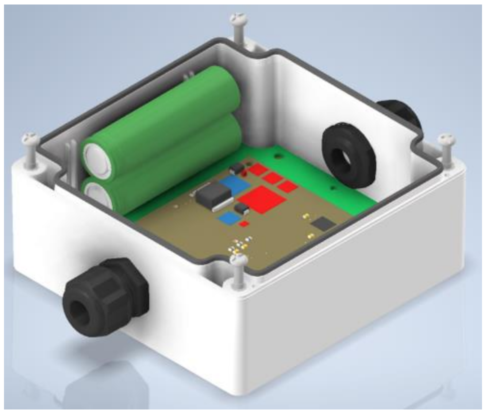
```
        console.println("actual probe 2 response " + String(temp));
        hasProbe1 = true;
      }
    }

    if (!hasProbe1){
      probe1 = NULL;
      console.println("Does not have DS18B Probe 1");
    }
    else{
      console.println("Has DS18B Probe 1");
      ioConfig.GPIO1Config = GPIO_CONFIG_DBS18;
      if (ioConfig.GPIO1Name == "")
        ioConfig.GPIO1Name = "Digital Temperature - Port 1";
      ioConfig.GPIO1Scaler = 1;
      ioConfig.GPIO1Zero = 0;
      ioConfig.GPIO1Calibration = 1;
    }

    probe2 = new DallasTemperature(new OneWire(IO2_PIN));
    retryCount = 0;
    while (retryCount++ < 5 && !hasProbe2)
    {
      float temp = probe2->getTempFByIndex(0);
      if (!isnan(temp) && temp > -50.60f)
```



```
  configureConsole();
  writeConfigPins();
  determineSensorConfiguration();

  console.registerCallback(handleConsoleCommand);
  welcome(TEMP_SNSR_SKU, FIRMWARE_VERSION);

  String btName = "NuvIoT - " + (sysConfig.DeviceId == "" ? "Temp Sensor" : sysConfig.DeviceId);

  BT.begin(btName.c_str(), TEMP_SNSR_SKU);

  sysConfig.WiFiSSID = "Collins";
  sysConfig.WiFiPWD = WIFIPW;

  wifiMgr.setup();

  ledManager.setup(&ioConfig);
  ledManager.setOnlineFlashRate(1);
  ledManager.setErrFlashRate(0);

  probes.configure(&ioConfig);
}
```

Test Plan #4 – Enclosure

A CAD design was initially developed for the enclosure and was to be 3D printed to fit the device. However, the enclosure needed to be IP68 rated to resist harsh conditions and environments, so ultimately, the enclosure was ordered, and we made the cutouts for the temperature sensor, LEDs, and charging ports. The enclosure is resistant to harsh conditions such as vibrations, dust, cold, heat, etc. It encloses the entire device and can be surface mounted to the equipment.



Test Plan #5 – Mobile App

The temperature from the thermistor is sent back to the server using the NB-IoT (Narrowband Internet of Things) technology. Low-power wide-area network realm is used to allow for private wireless network. The ESP32 microcontroller comes with built-in Wi-Fi and Bluetooth so we used the built-in Wi-Fi to connect to the internet. We used Bluetooth and firmware to perform final bits of configuration in order to send data back to the server. We attempted to adjust the already created react-native mobile app by *Software Logistics* to work with our board and microcontroller but had challenges and would need a 3rd part software firm. After testing, the device registers itself with the back-end application and is ready to use and send data. When the Bluetooth application doesn't work, we can configure the parameters needed in the firmware.

Summary

The IoT device operates and meets all developed requirements after completing final PCB design, device enclosure, firmware, and the associated NuvIoT app. Utilizing all test plans for every aspect of the project helped us to develop the device properly and ensure all components and the mobile app were interfacing correctly. The test plans also allowed us to realize when we were falling short of some of our goals and requirements, and when errors were occurring that we may never had noticed without proper testing. Test plan #1 was vital to the successfulness of our product, as the PCB and the temperature sensor were the backbone of our device. While several revisions of the PCB had to be developed, the final design allowed the device to function properly with all requirements being met. We struggled with the development of the mobile application, as it used react-native and C code, especially when it came to the Android side of the programming. However, we were ultimately able to merge all aspects of the device and the testing involved to overcome the challenges we faced, yielding a final working product ready to be implemented in the real world.