engineering proceedings

MDPI

Proceeding Paper

Predictive IoT Temperature Sensor †

Ali Elyounsi and Alexander N. Kalashnikov *

Department of Engineering and Mathematics, Sheffield Hallam University, Sheffield S1 1WB, UK; a8013965@my.shu.ac.uk

* Correspondence: a.kalashnikov@shu.ac.uk

† Presented at 9th International Electronic Conference on Sensors and Applications, 1–15 November 2022; Available online: https://ecsa-9.sciforum.net.

Abstract: Temperature sensors are widely employed in control systems that maintain required temperature in a vessel or container irrespective of the temperature changes in the outer environment. However, limited power of the heater/cooler (the plant of the control system) might lead to uncomfortable or even inacceptable deviations from the required temperature. This behaviour can be mitigated if the control system can have access not only to the present temperature in the vessel but also to the forecasted environmental temperature. This situation occurs, among others, at industrial vessels that require elevated temperatures during their operation but shut down out of hours. To start heating these to the required temperature at the beginning of a working shift wastes processing time until the required temperature is reached. It is more productive to turn on heating in advance in order to get the vessel ready on time. In order to achieve fully autonomous automatic operation, the sensor should have some intelligence and access to the temperature forecast, which can be provided over the internet. Both these requirements can be met by employing a WiFi enabled microcontroller. We present development of a predictive IoT temperature sensor based on the ESP32 microcontroller, which uses internet service to get time and weather forecast, and upload temperature logs to a cloud server for convenient remote access and storage.

Keywords: predictive temperature management; IoT sensor

Citation: Elyounsi, A.; Kalashnikov, A.N. Predictive IoT Temperature Sensor. *Eng. Proc.* **2022**, *4*, x. https://doi.org/10.3390/xxxxx

Academic Editor: Francisco Falcone

Published: 1 November 2022

1. Introduction

Temperature sensors are used for a wide variety of industrial, scientific, medical and domestic purposes, and they differ by design and/or operating principles to better suit their given application. The global market size for these sensors was estimated at 6.3 billion USD in 2020 with a projected annual growth of 4.8% to 2027 [1]. They are frequently employed in feedback control systems in order to maintain the required temperature of an object irrespective of the temperature changes in outer environment. For well insulated objects the power of the heater/cooler does not need to be very high as the object can be brought to the required temperature without significant heat losses albeit slowly. However, the associated time delays might lead to, for example, production losses if the object's temperature is outside of the range acceptable for a manufacturing process. If the heating/cooling requirements are known or/and can be predicted in advance, the temperature of the object can be managed much more efficiently.

Recent advances in Internet technologies, infrastructure and services made it possible to develop automated management systems, which provide the level of intelligence that cannot be achieved by using low cost computing at the edge alone. These services, in addition to already ubiquitous cloud storage, include, i.a., Internet-of-Thigs (IoT), Industrial Internet-of-Things (IIoT), time, weather forecast, mapping etc servers. As the result, it became possible to engineer control systems with features, which would be economically unfeasible a few years ago because of their high capital and running costs. This paper presents a detailed account of development of a prototype industrial control system,

featuring a single WiFi-enabled microcontroller unit (MCU), which extensively uses Internet services in order to cut off unnecessary energy use and associated costs for a process tank heater. The reported development was initiated by a Sheffield area-based enterprise of small/medium size that qualified for development support under the auspices of the Digital Innovation for Growth programme [2]. The company provides electrochemical processing services to their customers, which involves the use of heated water tanks. Many processes must be conducted at elevated temperatures to ensure conformity, consistency, and quality of outcome. Appropriate heating of some water tanks requires many hours and should be completed by the start of a working day to avoid any waiting time.

At present, the company uses the setup presented in Figure 1. A domestic timer is set at the end of a working day to switch on the heater after a guesstimated delay. When the in-tank temperature exceeds that required for the process to be conducted the next working day, the thermal limit switch switches the heater off. The tank cools down until the temperature crosses the lower threshold of the thermal switch, and the heater starts to operate again.

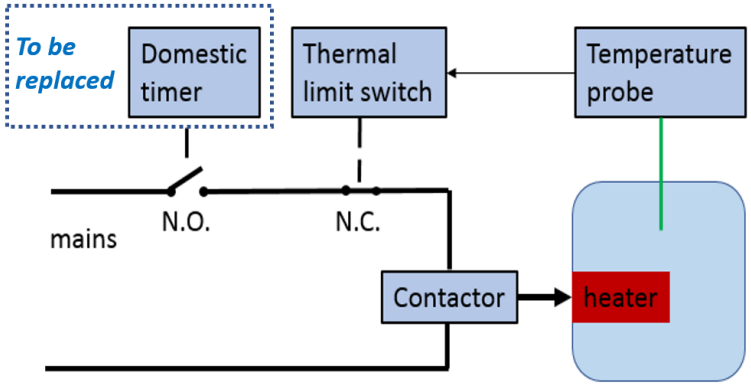

Figure 1. The existing control system and its modification for intelligent management.

This arrangement generally results in the required temperature by the start of the working day, but because it relies on the guesstimated delay, it can be hit or miss. If the delay is set with a spacious margin that ensures the tank's readiness, a substantial amount of energy can be wasted (Figure 2). If the delay is set with the view to minimise energy waste, production delays may be encountered.

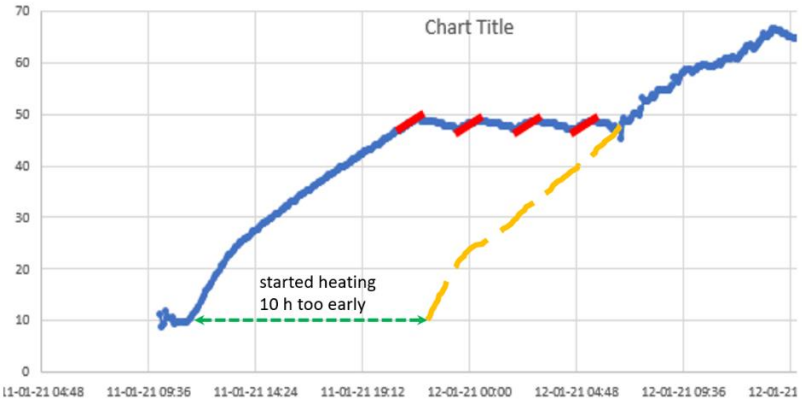

Figure 2. A typical heating curve with the heater switched on too early. Red bars show the time intervals when the heater had to be switched on to maintain the required temperature after the tank cooled off naturally, which wasted energy.

This project aims to develop an automatic heater control system that would achieve the required heated tank temperature at the start of a working day without excessive energy consumption by utilising ambient temperature forecast.

2. Project Requirements and Considerations

As there are approximately ten heated tanks around the company's premises that must be controlled, the customer requested development of an independent controller to simplify trial testing and deployment. The controller needed to save the temperature logs in the internet cloud and onto an SD card, and it needed to be able to access internet weather forecasts via the company's Wi-Fi access point. It also needed to be made from commercially available off-the-shelf components that do not require the manufacture or assembly of custom electronic boards. Last, the controller needed to be encased in a material that was flexible and expandable.

Safety concerns: the heater should be switched off if the temperature exceeds requirements or if the controller loses power.

Security concerns: temperature logs have no notable value to the company or to any actors with malicious intent. For this reason, operating via the company's password-protected access point is sufficient. The Wi-Fi credentials can be set externally after first power up by using a suitable Wi-Fi password manager that is then stored in the controller's RAM. However, because the company's premises are secure and the credentials are known to onsite personnel, the option to permanently store the credentials in the flash memory of the controller was instead selected.

3. Selection of the Hardware

Temperature measurements can be conducted by using a variety of sensors; thermocouples and resistance temperature detectors (RTDs) are the most common in industrial environments. The former has an extended temperature range but lower sensitivity. For this reason, and because the customer had previously installed RTDs on-site, the Pt100 RTD sensor was selected. It requires glue electronics to communicate data to a microcontroller using either analog voltage or a digital interface. The latter option requires more wiring but is considerably more robust and resilient to noise and was thus considered preferable. We selected an Adafruit Pt100 RTD temperature sensor amplifier [3], which eases use of the MAX31865 integrated circuit that is dedicated to RTD handling [4]. Additionally, we used a DS18B20 digital temperature sensor [5], a cheaper alternative that can be connected to a microcontroller without use of glue electronics.

Implementation of a Wi-Fi connection to obtain the weather forecast could be done by using a microcontroller with a Wi-Fi gateway that is controlled either through use of AT commands or by acting as a serial to the TCP converter. However, there is also a well-established line of Wi-Fi-enabled systems-on-chip (SoCs) that is manufactured by Espressif and allows for the use of a single component for both the control and Wi-Fi connection. Although the older generation ESP8266 series is more established and generally cheaper to deploy, the more recent ESP32 series offers significantly enhanced capabilities that are useful for prototyping and low volume replication [6].

M5Stack Basic Core [7] was selected because it provides access to a variety of ESP32 pins and allows for easy expansion by combination of the stackable modules. In particular, we connected the output of the Adafruit PT100 RTD temperature sensor amplifier to the SPI pins of the ESP32 and the data pin of the DS18B20 sensor to an ESP32 GPIO pin. Both sensors were powered by the 3.3 V source available on M5Stack Basic Core.

4. Selection of the Development Environment

ESP32 can be programmed using a variety of toolchains, including the Espressif IoT Development Framework ESP32-IDF (C/C++ compiler and linker), the Arduino environment, and MicroPython, which becomes increasingly popular because its interpreting

nature does not require full code re-compilation after every change. Arduino environment was selected because of availability of example codes, which covered all the identified needs of this project. This choice was eventually confirmed by the straightforward process of searching for solutions and workarounds when an added code snippet did not integrate well with the developed firmware.

5. Selection of the IoT Service Provider

Currently, the Message Queuing Telemetry Transport (MQTT) protocol seems to be the most used for Internet of Things (IoT) and Industrial Internet of Things (IIoT) applications [8]. It features a (cloud/LAN) broker that listens constantly for packets from data producers, stores them, and then serves data consumers upon their request. This allows both the producers and the consumers to experience disconnection or a loss of power without losing data. A cloud broker eliminates the need for maintaining a local one, which is convenient for a user without a dedicated infrastructure.

Despite the large number of Google search results for IoT service providers, most proved incompatible with the requirements of this project. Some were too expensive, at the yearly cost of £500 or more for commercial applications (Blynk, Thingspeak, Google Cloud, Particle IoT, Oracle IoT, Thinger, Ubidots, Pubnub, and IBM Watson to name a few). Other options did not advertise a transparent pricing structure (e.g., email communication was required for getting a quote from myDevices), some top providers would be unlikely to help with development due to the low potential income stream (Microsoft Azure IoT, Amazon IoT Core), some convoluted the process by splitting their operations between two entities (i.e., Dweet for data producers and Freeboard for data consumers), and one quite promising option went out of business (Phant).

In fact, we selected Adafruit IO [9] from the outset, as it ticked all the right boxes: easy setup, capable free-tier account, $10 monthly fee for commercial use, extensive documentation, and a support forum. Later, when we encountered difficulties, we searched for alternatives but found none. Throughout our use of Adafruit IO over several months, we did not lose a single datum, observed 100% server uptime, enjoyed their online Graphical User Interface (GUI) editor, and easily located relevant documentation when needed.

6. Firmware structure and functionality

The firmware development started by exploring examples and then integrated the selected code snippets from the well-established **Adafruit_MAX31865** (for RTD) and **OneWire/DallasTemperature** (for DS18B20) Arduino libraries. The Network Time Protocol (NTP) servers were accessed based on the code snippets from [10]. For weather forecasting we used a free service called Datapoint that is provided by the UK's Meteorological Office (Met Office) to businesses in the UK (it requires registration to get an access key and location code) [11]. In order to prevent the device from hanging, we employed the ESP32 watchdog timer based on the code snippets from [12]. Adafruit IO code was integrated at the final stage, after familiarisation with the provided examples. After the relatively straightforward process of getting the code to work, the user interface (a.k.a. dashboard) was easily developed through use of Adafruit's online dashboard editor (Figure 3).

Figure 3. Prototyped Adafruit IO dashboard showing historical graphs on the left, and present values and manual control switch on the right. Sampling rate was approximately 20 s. The graphs shown cover the interval of 30 days, which is user selectable. By hovering a mouse pointer over any graph, the user can observe numerical values of the data of interest.

As using the Adafruit IO Arduino API resulted in some complications, the Adafruit IO REST API (via POST and GET HTTP requests) were employed instead. The developed code was shared on the Adafruit IO support forum [13].

The device, when operating at full throttle, consumes around 0.11 A of current. During the light sleep, M5Stack consumes less than 0.01 A, which results in the reduction of power consumption by over ten times, as sampling temperature and reporting results takes less than 5 s.

The **loop** Arduino function, therefore, takes temperature samples from the two temperature sensors, sends the readings to the cloud, displays them on the local M5Stack colour LCD display, checks the position of the manual switch on the dashboard, determines if it is necessary to update the RTC and weather forecast, clears the watchdog timer, and then enters light sleep mode.

The time to switch the heater on is calculated by considering the heat losses through the surface and walls of the tank, rated heater power, and the difference between the tank's required and forecasted ambient temperatures.

The prototyped device is presented in Figure 4.

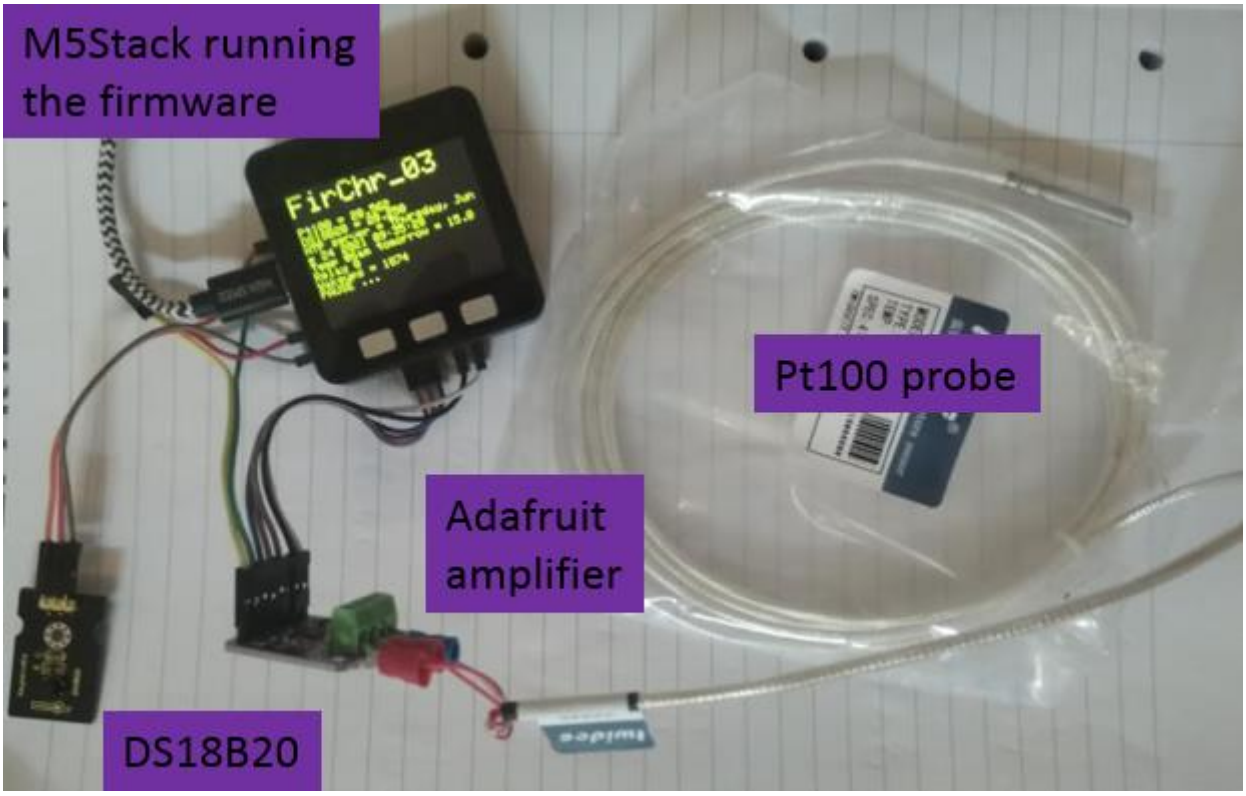

Figure 4. Controller's prototype at the testing stage.

7. Summary and Conclusions

We detailed development of a prototype intelligent control system, including considerations for relevant hardware, development environment, Internet services used (NTP, weather forecast, cloud MQTT servers) and results of the development.

In particular, the system made use of the following technologies and services:

NTP servers for getting accurate time references;

Adafruit IO's MQTT servers, acting as a broker between the data producer (the ESP32 MCU) and the data consumer (the end user);

Adafruit's IO's dashboard and layout editor for designing the graphical user interface for the end user;

UK Met Office's Datapoint for getting location and time specific weather forecast;

Adafruit's Arduino application programming interface calls were replaced with more robust HTTP's GET and PUT commands.

The required hardware includes only parts, available off-the shelf.

The prototype allows the water tanks achieve and maintain the required temperatures, excess energy consumption is curtailed, and absolutely no guesstimation is involved.

Author Contributions:

Funding:

Institutional Review Board Statement:

Informed Consent Statement:

Data Availability Statement:

Acknowledgments: Ali Elyounsi gratefully acknowledge support for his studies from Libyan Embassy.

Conflicts of Interest:

References

1. Markets and Markets. Temperature Sensor Market by Product Type (Thermocouples, RTDs, Thermistors, Temperature Sensor ICs, Bimetallic, Infrared, and Fiber Optic Temperature Sensors), Output, End-User Industry, and Region—Global Forecast to 2027. 2020. Available online: https://www.marketsandmarkets.com/Market-Reports/temperature-sensor-market-522.html (accessed on September 2022).
2. Digital Innovation for Growth (DIfG). Available online: https://www.shu.ac.uk/business/support/start-ups-smes/digital-innovation-for-growth (accessed on September 2022).
3. Adafruit PT100 RTD Temperature Sensor Amplifier. Available online: https://www.adafruit.com/product/3328 (accessed on September 2022).
4. MAX31865 RTD-to-Digital Converter (Datasheet). Available online: https://datasheets.maximintegrated.com/en/ds/MAX31865.pdf (accessed on September 2022).
5. DS18B20 Programmable Resolution 1-Wire Digital Thermometer. Available online: www.maximintegrated.com/en/products/sensors/DS18B20.html (accessed on September 2022).
6. ESP32 vs. ESP8266—Pros and Cons. Available online: https://makeradvisor.com/esp32-vs-esp8266, accessed Sept 2022.
7. ESP32 Basic Core IoT Development Kit (Product Page). Available online: https://shop.m5stack.com/collections/m5-core/products/basic-core-iot-development-kit (accessed on September 2022).
8. MQTT: the Standard for IoT messaging. Available online: https://mqtt.org/ (accessed on September 2022).
9. What is Adafruit IO?. Available online: https://learn.adafruit.com/welcome-to-adafruit-io/what-is-adafruit-io (accessed on September 2022).
10. Getting Date and Time form NTP Server Using ESP32. Available online: https://lastminuteengineers.com/esp32-ntp-server-date-time-tutorial/ (accessed on September 2022).
11. Met Office Data Point. Available online: https://www.metoffice.gov.uk/services/data/datapoint (accessed on September 2022).
12. How to enable hardware WDT on ESP32 using Arduino IDE. Available online: https://iotassistant.io/esp32/enable-hardware-watchdog-timer-esp32-arduino-ide/ (accessed on September 2022).
13. Using POST to Add Data Points to a Feed. Available online: https://forums.adafruit.com/viewtopic.php?f=56&t=177055 (accessed on September 2022).