ABSTRACT

Accurate temperature measurements are needed to monitor and reduce energy use and greenhouse gas emissions in building, industrial, and other processes. This paper describes a temperature sensor instrument, named TempSense, developed for classroom demonstrations of thermal phenomena. The instrument can monitor, record, and visualize data from multiple temperature sensors using a K-type amplifier and thermocouples. It has an accuracy of ±2 deg C from -200 to +700 deg C and an accuracy of ±6 deg C from -270 to +1372 deg C. The instrument uses a USB bridge and the Python language to interface with open-source thermocouple amplifiers without the need for byte-level programming. Its design allows novice experimenters to inspect the build and operation and even customize it to their needs. TempSense can be extended via Python libraries to provide internet communication, visualization, and analysis.
(1) OVERVIEW

METADATA OVERVIEW
Main design files: https://doi.org/10.5281/zenodo.4279570
Target group: instructors, students, engineers
Skills required: through-hole soldering – easy; computer terminal commands – medium
Replication: Instrument has been installed by another educator
See section “Build Details” for more detail.

INTRODUCTION
Temperature measurements and time series data are used to support efficient uses of energy and thus combat climate change. Collecting these data at a large scale requires wide use of temperature sensor technologies in building, industrial, and other processes, as well as sufficient understanding of the instruments and data collection methods. In my conversations with energy practitioners, I have learned that these instruments are underutilized and often not installed properly, often making the data they collect impossible to interpret. To address these issues, skills such as collection and interpretation of temperature time series data should be included in science curricula and professional training programs. To aid in this instruction, I developed a temperature sensor instrument, called TempSense, as a classroom demonstration tool to help students relate their physical observations to real-time visualizations and digital data sets. The tool is well-suited for classroom demonstrations and laboratory use, as well as for practical temperature logging applications.

High-quality, rugged, and reliable multi-channel temperature sensors are relatively costly (300–500 USD) and not easily extended, thus creating a niche user base for this instrument. The Omega HH1385, a 4-input thermometer with a USB interface for Microsoft Windows and data logging capabilities, sells for 425 USD (Omega, 2020). The Extech SDL200, a 4-channel data logger for thermocouples with data logging to an SD card, sells for 360 USD (Extech, 2020). A set of four inexpensive thermocouples for each of these would increase the cost by 40–50 USD. Moreover, the Omega and Extech instruments have no multiplatform path to stream data; the PASCO (2020) tool offers streaming, but the software platforms used are proprietary. The Pax Instruments T400 device, a 4-channel standalone logger using thermocouples with a USB interface and open hardware and software, sold for 175 USD but was discontinued (Pax Instruments 2020). Physics education researchers designed a similar system for education called Termocrono, but it is not open hardware (Michelini, Santi, and Stefanel, 2010).

The materials cost for the TempSense temperature sensor described in this paper is below 150 USD. Because it is designed for stationary monitoring and uses minimal, open-source circuitry, no rugged case is needed, thus reducing the cost. Use of a connected host computer eliminates the need for battery power. However, it requires Python, USB library installations, and an environment variable configuration, which can be challenging for novices.

OVERALL IMPLEMENTATION AND DESIGN
To create a low-cost, customizable, and open-source temperature sensor that is accessible and useful for novices, the hardware design must be affordable, extensible, and approachable. Several design aspects of the proposed tool enhance its affordability. For example, using a connected host computer and USB bridge (Future Technology Devices International Limited, 2020) removes the need for a microcontroller or battery power. Because the TempSense instrument is designed for stationary monitoring, no rugged case is needed. A minimal circuit board area, widely available thermocouples, and an open-source thermocouple amplifier circuit board also reduce costs (Maxim Integrated, 2020; Adafruit, 2020c), and the through-hole soldering assembly is performed by the end user. Figure 1 illustrates the assembled instrument and bare printed circuit board.
The instrument platform uses Python in a single codebase on the host computer, thus making it highly extensible. The CircuitPython (2020) libraries for reading the thermocouples do not require byte-level data manipulation. The USB bridge replaces the microcontroller, so the only instrument code is a Python script on the host computer. The CircuitPython libraries are compatible with mainstream Python libraries, allowing a rich set of graphing, communication, and internet libraries to be integrated with the instrument. Data can be sent to a database or setpoint; maximum and minimum alarms can be sent by email or text message; and real-time visualizations can be constructed.

Figure 2 displays a block diagram of the software libraries used. The USB bridge chip translates between the USB commands for the host computer and serial peripheral interface (SPI) commands for the thermocouple amplifiers. The libusb, pyusb, and pyftdi libraries translate between Python commands and USB data packets. The CircuitPython and Blinka libraries provide a simple Python interface to the thermocouple amplifiers.
Several design features of the proposed temperature sensor instrument allow learners to understand, extend, and customize device operation. Using Python, which is a widely-taught and approachable language, makes it easier for coding novices to engage in this process. The instrument code is written in a style that favors clarity over compactness, performance, or idioms. Instrument code that generates text output only is included as a minimal working example to show how the instrument is used. Instrument code with real-time graphing using the Bokeh (2020) visualization library can be customized once learners understand the basic operation of the instrument. To help students verify the signal paths of the thermocouples, the traces used to select each amplifier chip for communication are clearly labeled on the front of the circuit board and the circuit schematic, as shown in Figure 3.

**EDUCATIONAL PRINCIPLES**

The creation of this tool was motivated by the need for an affordable and easy-to-use sensor for collecting real-time temperature measurements and data sets. STEM education research shows the value of having students create thermal predictions and compare them against authoritative physical observations (Prince, Vigeant, and Nottis, 2016; Kean et al., 2008). When these comparisons highlight misconceptions, the student can reflect and update their mental models to reconcile their predictions and observations. While teaching energy concepts to undergraduate students, I observed that they tend to conflate concepts related to the sensible heating of water with the latent heats of evaporation and freezing. I thus created a lesson asking students to predict the rates of ice melting, water heating, and evaporation based on a constant input of thermal energy from a heater while also observing the temperature of the ice and water mixture in real time. Lacking a simple sensor with real-time graphing and temperature data logging, I developed the TempSense instrument.

The comprehensible and extensible nature of the instrument can enhance students' and other novices' data literacy. The first iteration of the instrument used a client/server architecture with a host computer running Python and a microcontroller running CircuitPython. Because the FT232H USB bridge has a CircuitPython library, the instrument’s code can be contained completely on the host computer, which facilitates adaptation and upgrades. The included software is intentionally simple with instructional comments explaining the basic structure of the code. The readability of Python and the simplicity of the code allows customization of graph elements (e.g., labels, dots, lines, and colors) and offers an approachable entry into data collection instruments and data-science concepts.
(2) QUALITY CONTROL

SAFETY

During assembly, it is important to follow all necessary safety precautions for soldering, including using caution around hot soldering iron tips, ensuring adequate ventilation of flux fumes, wearing eye protection, and using low-toxicity lead-free solder. During use, caution should be exercised to avoid burns and other injuries when placing thermocouples.

CALIBRATION

I validated and characterized the instrument by calibrating ice-point and boiling-point temperatures of water at an elevation of 9 meters using a single sensor. The ice-point calibration was performed using an ice-water mixture in a 500-ml beaker. During calibration, 100 data points were recorded at 5-second intervals using a shielded thermocouple. The thermocouple was placed near the top of the container where both ice and water were present. The observed mean of the 100 samples was 2.52 deg C, and the standard deviation was 0.072 deg C. The boiling-point calibration was performed with 500 ml of boiling water on a laboratory hot plate. During calibration, 100 data points were recorded at 1-second intervals using a shielded thermocouple. The mean reading was 98.7 deg C with a standard deviation of 0.15 deg C.

These means are consistent with a typical thermocouple accuracy of 2.2 deg C, and the deviations are consistent with the MAX 31855K amplifier resolution of 0.25 deg C (Maxim Integrated, 2020). The MAX 31855K amplifier datasheet specifies an accuracy of ±2 deg C from −200 to +700 deg C and an accuracy of ±6 deg C over the wider range of −270 to +1372 deg C. For users wanting higher control over calibration, the cold-junction compensation code can be accessed in the MAX 31855 CircuitPython library (Adafruit, 2020a). The raw data for these calibrations are available at https://doi.org/10.5281/zenodo.4279547.

GENERAL TESTING

To demonstrate the operation of the instrument, I measured the temperatures along a conductive bar that was heated at one end. These temperature measurements demonstrated a spatial temperature difference that increased over time, similar to measurements to determine temperature differences and energy efficiency of a wall or window.

To create the temperature differences, I heated a 2 mm × 13 mm × 200 mm strip of aluminum using a typical laboratory hot plate. I attached thermocouples along the aluminum strip at 5-cm intervals using high-temperature polyimide tape. A clamp was fastened to maintain contact between one end of the aluminum strip and the hot plate. One thermocouple was placed in the thermal contact area, and the other three were placed beyond the area of the plate. Temperature measurements were taken at 10-second intervals. Figure 4 shows the apparatus.

Figure 4 Experimental apparatus to demonstrate temperature difference evolution over time.
The results indicated the temperature gradient along the aluminum strip, with the area in contact reaching recording over 270 deg C and the farthest thermocouple over 100 deg C. Figure 5 shows the real-time graphing interface. The raw data for this demonstration are available at https://doi.org/10.5281/zenodo.4279547.

(3) APPLICATION

USE CASES

This instrument is suited for multi-channel temperature observations where data logging and real-time visualization are desired. The real-time visualization capacity fits classroom activities where students make predictions and compare them to observations. The visualization library, Bokeh, includes pan and zoom capabilities so that instructors and students can inspect the data while logging. This paper describes use of this instrument to teach undergraduate students how to log heating curve data for melting, heating, and boiling water. I also have used it in a building science lesson to demonstrate transient and steady-state behavior of interior and exterior temperatures of a small enclosure heated by an internal heater. In another lesson, I demonstrated fan-driven evaporative cooling of water by simultaneously displaying the water temperature and air temperature. Although these lessons were designed for undergraduate students and focused on quantitative prediction, similar instruments (e.g., the Termocrono instrument by Michelini, Santì, and Stefanel (2010)) have been used to teach kindergarteners about sensory experiences, showing their potential for all ages.

Another use case for the instrument is as a simple model to understand data collection technology and methods. At the data presentation level, for example, students can customize the labels and colors of the graph for a particular experiment by modifying the Python code. The instrument also shows the chain of temperature representations from the sensor to digitization to the graph or text output. Students could trace and label these representations
as an exercise to better understand the data collection process. At a conceptual level, students can easily observe the instrument’s representations (voltage, number, graph distances), which demystifies the internal workings of the instrument and thus increases STEM competency, particularly as it relates to data science.

**REUSE POTENTIAL AND ADAPTABILITY**

This instrument can be used in situations where temperature observations are needed and a connected host computer can be used. The ability to change the logging frequency and real-time graph styles encourages reuse. Any K-type thermocouple can be used, allowing adaptation to different environments.

All instrument code resides on the host laptop or desktop computer, so it is straightforward to adapt the instrument’s functions by extending the code. Data could be logged directly to an Excel or Google spreadsheet using available open-source libraries like gspread (Burnashev, 2020) and xlwings (2020) and sent to cloud-hosting services like Adafruit IO (Adafruit, 2020d). Derived quantities (temperature difference, heat transfer) could be displayed in real time by extending the code and adding graph or text outputs. The CircuitPython code also could be adapted (without the USB bridge) to other platforms, such as single-board computers with SPI or I2C interfaces (Raspberry Pi, BeagleBone) or embedded devices (Feather, Grand Central), providing a much more portable device.

**(4) BUILD DETAILS**

**AVAILABILITY OF MATERIALS AND METHODS**

The instrument uses relatively inexpensive parts from well-established vendors. The key electronic parts, the USB bridge and thermocouple amplifier boards, are designed and sold by Adafruit as open-source hardware. The connecting sockets and thermocouples are available from Digikey. The printed circuit board is available from OSHPark (2020) and other vendors by sending the KiCad printed circuit board file. The assembly requires a standard soldering iron and solder, which are widely available.

**EASE OF BUILD**

The assembly of this instrument requires soldering skills and equipment. Novice soldering skills are sufficient because the pins and connectors use a forgiving 0.1-inch pitch. To install the computer software libraries, an understanding of the command line and scripts, library dependencies, and package installation is required. The repository contains detailed instructions and troubleshooting for this process. Once these software library dependencies are installed, the instrument is run using a single terminal command.

**OPERATING SOFTWARE AND PERIPHERALS**

The instrument can be run on any operating system that has a Python installation. The software needed to operate the instrument is included in the repository. One Python script contains commands for simple text output to the terminal, and another script contains the graphing interface. An included shell script configures the Python environment, sets a shell environment variable, and launches the Python script. This software can be easily extended because all source code is stored in text files on the host computer.

**DEPENDENCIES**

The instrument requires a host computer (Linux, OS X, or Windows) with a Python installation. The Python library dependencies are listed in a format for use by the Anaconda/Conda package and environment manager (Anaconda, 2020). The printed circuit board files are in KiCad format.

**HARDWARE DOCUMENTATION AND FILES LOCATION**

*Archive for hardware documentation and build files*

- **Name:** TempSense Build Instructions
- **Persistent identifier:** https://doi.org/10.5281/zenodo.4279570
This archive contains the modifiable build files and software at the time of publication. The current version is available in a GitHub repository at https://github.com/dsoto/TempSense.

(5) DISCUSSION

CONCLUSIONS

This manuscript presents a multi-channel, low-cost, multi-platform, extensible temperature sensor featuring real-time data display and data logging. This TempSense instrument has been used in several classroom demonstrations where students make predictions and compare them to observations. A water ice-point and boiling-point calibration was used to verify the operation of instrument, and a differential heating demonstration was performed. This instrument uses a USB-bridge architecture with all code on the host computer, which creates possibilities for other USB-connected instruments. The instrument is intended to help instructors teach digital data collection and visualization to strengthen students’ learning.

FUTURE PEDAGOGICAL WORK

Future work is planned to help instructors use this instrument to enhance learning. Research shows the benefit of prediction and observation activities, but the time and resource investment for new lessons and hardware is significant. On the technical side, I will continue to refine the installation resources to help instructors with less computer experience install the instrument software libraries efficiently. To align the use of the instrument with learning objectives, I will continue to develop and publish lessons, questions, and activities for different thermal concepts (e.g., phase transitions, thermal transport, evaporative cooling). These lessons will include prediction activities designed to provide real-time visualizations of temperature and to dispel common misconceptions. As such, the lessons could be adapted to Next Generation Science Standards (2020) for teaching thermal energy concepts to middle and high school students.

In addition to temperature concepts, general data science concepts and techniques can be taught with this instrument. For example, students can inspect the recorded data points and compare them other data. They can observe the relationships between rows and columns of data, where rows usually indicate simultaneous observations and columns indicate a specific type of observation. For studying the concepts of slope and area as they relate to time series data, the instrument’s data files can be used to perform pairwise differences and trapezoidal integrations and relate them to thermal energy flows. Lessons on these concepts create a foundation to understand other methods of data collection in physical and even social sciences.

FUTURE TECHNICAL WORK

It is possible to extend this CircuitPython and USB bridge architecture to any sensor with an SPI or I2C interface. It is therefore also possible to create USB instruments that do not require a separate microcontroller, simplifying development and customization. CircuitPython maintains an extensive list of libraries to interface with SPI and I2C devices (Adafruit, 2020b). For example, combining a USB bridge with an SPI or I2C analog-to-digital converter creates a low-cost and easy-to-use data acquisition system.

Some SPI and I2C devices are incompatible with the USB bridge used for this project. For example, the Future Technology Devices International Limited FT232H USB bridge supports only two (mode 0 and 2) of the four SPI modes. Using another USB bridge, such as MCP2210 (Microchip, 2020), to support all four SPI modes would allow a wider range of sensors. Of particular interest is the MAX 31865 amplifier for high-accuracy and precision resistance temperature detectors. MAX 31865 supports only SPI modes 1 and 3. Thus, it is incompatible with the FT232H. Other USB bridges use the USB Human Interface Device protocol, which could simplify the driver configuration and installation process. Future work could develop a version of this device that supports all 4 SPI modes and I2C for maximum compatibility with existing devices.
ACKNOWLEDGEMENTS

Alex Barajas and Thomas Swanson assisted in the original development of the code and hardware for this device. The students of the Energy, Management, and Design program at Sonoma State University were the inspiration for this instrument and gave feedback on its use in the classroom. Milo Mitchell at Orchard View School replicated the instrument and provided feedback on the installation instructions.

FUNDING INFORMATION

The Geography, Environment, and Planning department at Sonoma State University funded some of the development hardware. The MESA program at Sonoma State University provided student support. This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

COMPETING INTERESTS

The author has no competing interests to declare.

AUTHOR AFFILIATION

Daniel Soto  
Sonoma State University, US

REFERENCES

Adafruit. (2020a). CircuitPython driver for MAX31855 Thermocouple Amplifier. Available at: https://github.com/adafruit/Adafruit_CircuitPython_MAX31855 [Accessed 1 Jun. 2021].

Adafruit. (2020b). CircuitPython Library Bundle Device Drivers. Available at: https://github.com/adafruit/Adafruit_CircuitPython_Bundle/tree/master/libraries/drivers [Accessed 1 Jun. 2021].

Adafruit. (2020c). Thermocouple Amplifier MAX31855 breakout board. Available at: https://www.adafruit.com/product/269 [Accessed 1 Jun. 2021].

Adafruit. (2020d). Welcome to Adafruit IO. Available at: https://io.adafruit.com/ [Accessed 1 Jun. 2021].

Anacrona. (2020). Anaconda Individual Edition Python Distribution. Available at: https://www.anacrona.com/products/individual [Accessed 1 Jun. 2021].

Bokeh. (2020). The Bokeh Visualization Library. Available at: https://bokeh.org/ [Accessed 1 Jun. 2021].

Burnashev, A. (2020). GSpread Google Sheets Python API. Available at: https://github.com/burnash/gspread.

CircuitPython. (2020). Available at: https://circuitpython.org/ [Accessed 1 Jun. 2021].

Extech. (2020). SDL200: 4-Channel Datalogging Thermometer. Available at: http://www.extech.com/products/SDL200 [Accessed 1 Jun. 2021].

Future Technology Devices International Limited. (2020). FT232H – Hi-Speed Single Channel USB UART/FIFO IC. Available at: https://www.ftdichip.com/Products/iCs/FT232H.htm [Accessed 1 Jun. 2021].

Kean, A., Self, B., Miller, R., and Hamilton, E. (2008). Identifying Robust Student Misconceptions in Thermal Science Using Model Eliciting Activities. ASEE Annual Conference & Exposition. Pittsburgh, Pennsylvania.

Maxim Integrated. (2020). MAX31855 Cold-Junction Compensated Thermocouple-to-Digital Converter. Available at: https://www.maximintegrated.com/en/products/interface/sensor-interface/MAX31855.html [Accessed 1 Jun. 2021].

Michelini, M., Santì, L., and Stefanel, A. (2010). Thermal Sensors Interfaced with Computer as Extension of Senses in Kindergarten and Primary School. Il Nuovo Cimento C, 33(3), pp. 171–179.

Microchip. (2020). MCP2210 USB to SPI Converter with GPIO (Master Mode). Available at: https://www.microchip.com/wwwproducts/en/MCP2210 [Accessed 1 Jun. 2021].

Next Generation Science Standards. (2020). Available at: https://www.nextgenscience.org/ [Accessed 1 Jun. 2021].

Omega. (2020). 4 Input Thermometer and Data Logger with USB Interface. Available at: https://www.omega.com/en-us/sensors-and-sensing-equipment/temperature/thermometers/p/HH1384-Series [Accessed 1 Jun. 2021].

OSHPark. (2020). Available at: https://oshpark.com/ [Accessed 1 Jun. 2021].

PASCO. (2020). “Temperature Sensors.” Available at: https://www.pasco.com/products/sensors/temperature [Accessed 1 Jun. 2021].
Pax Instruments. (2020). Pax Instruments T400 Temperature Datalogger. Available at: https://www.kickstarter.com/projects/1880050573/pax-instruments-t400-temperature-datalogger [Accessed 1 Jun. 2021].

Prince, M., Vigeant, M., and Nottis, K. (2016). Repairing Student Misconceptions in Heat Transfer Using Inquiry-Based Activities. Chemical Engineering Education, p. 50.

XLWings. (2020). Python for Excel. Available at: https://www.xlwings.org/ [Accessed 1 Jun. 2021].