QSD Monitoring Network

Measuring Temperatures

T. Barrett

July 2018

Measuring Temperatures

For sensing temperatures in the lab, it was decided to use thermocouples, since they are cheap, reliable, and allow a high range of temperatures to measured. This means that the same system can be used when baking out a vacuum chamber, which typically means reaching temperatures of up to 200 °C. Specifically, the most common thermocouple is the K-type, which has a range of around ($-200\,^{\circ}\mathrm{C} \rightarrow 1250\,^{\circ}\mathrm{C}$).

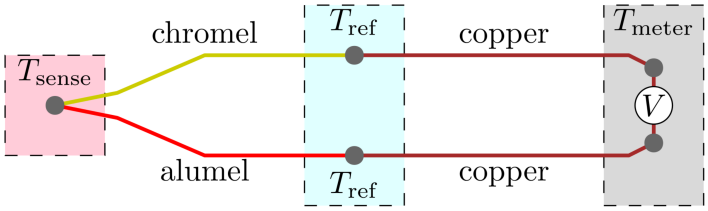


FIGURE 1: Schematic of a thermocouple. Two different metals are joined and positioned at the region to be sensed, T_{sense}. By knowing the temperature at the reference point, T_{ref}, the absolute temperature can be determined by appropriate conversion of the voltage, V. Image taken from [1]

A thermocouple, shown schematically in Fig. 1, relies on the *Seebeck Effect*, which describes the appearance of temperature-dependent voltage at the junction of two wires of different metals, converting thermal energy into electrical energy. For the K-type thermocouples, the positive lead is usually chromel (90% nickel, 10% chromium), and the negative lead alumel (95% nickel, 2% aluminium, 2% magnesium, 1% silicon). The voltage produced is very small (typically $\sim 1\,\mathrm{mV}$ at room temperature, with a sensitivity of $\sim 41\,\mu\mathrm{V}\,^{\circ}\mathrm{C}^{-1}$) and so must be amplified to a measurable value. In addition, the voltage must be linearised, along with being corrected for the cold-junction temperature. This is because the thermocouple is a differential device and so must be compensated with a reference junction of known temperature, in order to obtain an absolute value.

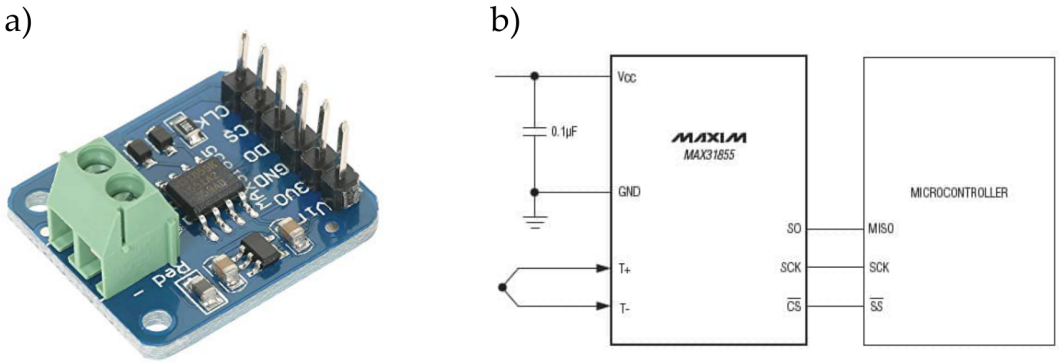


FIGURE 2: a) Adafruit MAX31855K assembled breakout board and b) connections schematic to the chip itself.

A convenient integrated circuit (IC) chip to use for this purpose is the MAX31855K from *Maxim*, which is optimised specifically for the K-type thermocouple, and whose schematic is shown in Fig. 2 b). This chip performs cold-junction compensation by measuring its internal

temperature acting as a reference, in addition to the hot junction. Subsequently, the chip also converts the thermocouple voltage into a 14-bit digital signal ready for readout by using an on-board analog-to-digital converter (ADC), and so doesn't require any analog inputs to work. The chip allows the temperature to be read out with a resolution of $0.25\,^{\circ}\mathrm{C}$, and exhibits and accuracy of $\pm 2\,^{\circ}\mathrm{C}$. Note in the datasheet that this IC has a typical conversion time of $\sim 70\,\mathrm{ms}$, during which time the cold and hot junction voltages are converted, along with some fault detection being performed. In practice, this limits the frequency with which temperatures can be measured to $\sim 10\,\mathrm{Hz}$, as one needs to wait at least until the conversion has finished before the next value can be read.

Communication with the IC (slave) is achieved using the serial peripheral interface (SPI) protocol, which requires connections to three digital I/O pins on an external microcontroller (master). The signals required are:

- **SCK** - *serial clock* - provides timing synchronisation between devices, and indicates when the next bit of data should be presented.
- **SS** (or **CS**) - *slave select* (or *chip select*) - typically pulled low by the master to indicate to the slave that it's time to read data.
- **MISO** - *master in slave out* - contains each bit of data sent from the slave to the master.

The MAX31855K can be purchased already on a breakout board (for example, from Adafruit here: `www.adafruit.com/product/269`), which already has the necessary $3.3\,\mathrm{V}$ regulator for the power supply, along with level-shifting between $3.3\,\mathrm{V}$ and $5\,\mathrm{V}$ for the various communication lines. This breakout board is shown in Fig. 2 a).

To interface between an Arduino microcontroller and the MAX31855K chip, it is most simple to use the *Adafruit MAX31855.h* library, available here:

`https://github.com/adafruit/Adafruit-MAX31855-library.`

In order to use the library, it is required to declare three digital output pins in the sketch, for CLK, CS, and DO. Then a thermocouple object should be created using the command:

```
1 Adafruit_MAX31855 thermocouple(CLK, CS, DO);
```

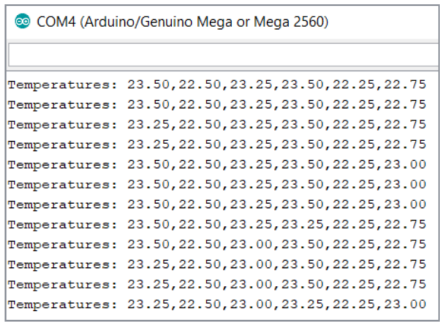
Finally, the temperature in celsius can be read from the newly created object using the following command:

```
1 T_value = thermocouple.readCelsius();
```

The CS pin allows multiple MAX31855K ICs to be connected to the same SPI bus (all using the same SCK and MISO pins), but with each having their own CS pins. Then, when a particular CS pin is pulled low, only the corresponding chip begins listening on the bus, and all others being high will ignore the master. A complete minimum working example (MWE) allowing the Arduino to read x6 thermocouples is shown in the section **Arduino Code** below, which requires x6 independent CS pins, along with a shared DO pin and a shared CLK pin. The output from

the Arduino IDE Serial Monitor is shown in Fig. 3 a), displaying a string of comma-separated temperature values being received over the serial port once per second. The time evolution of these temperatures after being saved to an external text file is shown in Fig. 3 b).

a)



b)

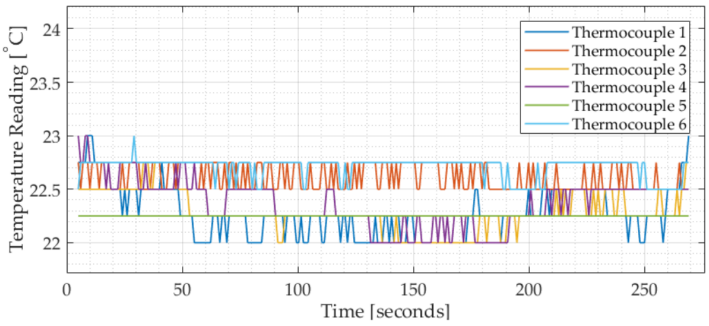


FIGURE 3: a) Six temperatures being received over the serial port, after being read from the MAX31855K ICs. b) Time evolution of received temperatures over several minutes.

Arduino Code

```
/* 01_MultipleThermocoupleRead
 * Code for measuring multiple temperatures using MAX31855 chips.
*/

#include<SPI.h>
#include "Adafruit_MAX31855.h"

#define DO  2 // Common MOSI pin
#define CLK 3 // Common serial clock pin

// Chip select pins
#define CS01  4
#define CS02  5
#define CS03  6
#define CS04  7
#define CS05  8
#define CS06  9

//THERMOCOUPLES
Adafruit_MAX31855 thermocouple01(CLK, CS01, DO);
Adafruit_MAX31855 thermocouple02(CLK, CS02, DO);
Adafruit_MAX31855 thermocouple03(CLK, CS03, DO);
Adafruit_MAX31855 thermocouple04(CLK, CS04, DO);
Adafruit_MAX31855 thermocouple05(CLK, CS05, DO);
Adafruit_MAX31855 thermocouple06(CLK, CS06, DO);
//

String payloadStr;
double temps[6]; // Buffer to hold thermocouple data

void setup() {
  Serial.begin(9600);
}

void loop() {

  // Read thermocouples
  temps[0] = thermocouple01.readCelsius();
  temps[1] = thermocouple02.readCelsius();
  temps[2] = thermocouple03.readCelsius();
  temps[3] = thermocouple04.readCelsius();
  temps[4] = thermocouple05.readCelsius();
  temps[5] = thermocouple06.readCelsius();

  // Construct string to print, using "0.0" if NaN is returned
  payloadStr = "";
  for (int i = 0; i < 6; i++) {
    if (isnan(temps[i])) {
      payloadStr = payloadStr + "," + "0.0";
    } else {
      payloadStr = payloadStr + "," + temps[i];
    }
  }

  Serial.print("Temperatures: ");
  Serial.println(payloadStr);

  delay(1000);

}
```