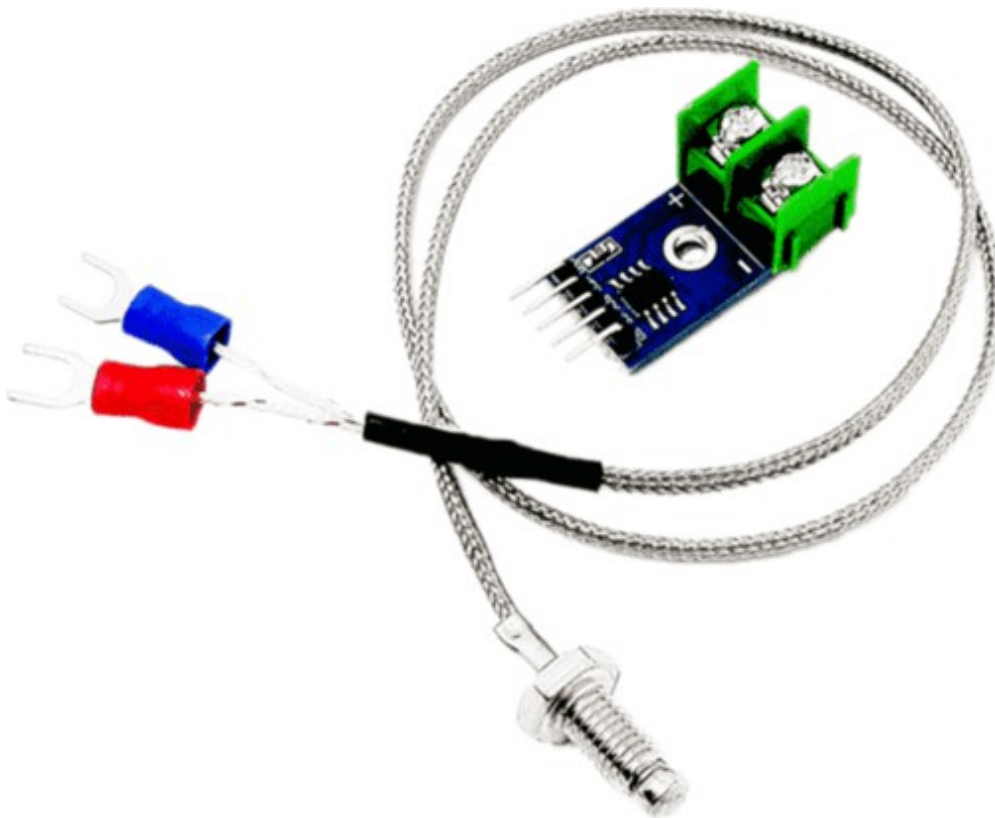


# AZ-Delivery

## Welcome to our website!

Thank you for choosing our **"MAX6675"** temperature sensor with **probe** from AZ-Delivery. In the following pages we will explain how to set up and use the device.

Have fun!



## Areas of application

Education and teaching: Use in schools, universities and training institutions to teach the basics of electronics, programming and embedded systems. Research and development: Use in research and development projects to create prototypes and experiments in the fields of electronics and computer science. Prototype development: Use in the development and testing of new electronic circuits and devices. Hobby and Maker Projects: Used by electronics enthusiasts and hobbyists to develop and implement DIY projects.

## Required knowledge and skills

Basic understanding of electronics and electrical engineering. Knowledge of programming, especially in the C/C++ programming language. Ability to read schematics and design simple circuits. Experience working with electronic components and soldering.

## Operating conditions

The product may only be operated with the voltages specified in the data sheet to avoid damage. A stabilized DC power source is required for operation. When connecting to other electronic components and circuits, the maximum current and voltage limits must be observed to avoid overloads and damage.

## Environmental conditions

The product should be used in a clean, dry environment to avoid damage caused by moisture or dust. Protect the product from direct sunlight (UV)

## Intended Use

The product is designed for use in educational, research and development environments. It is used to develop, program and prototype electronic projects and applications. The Sensor product is not intended as a finished consumer product, but rather as a tool for technically savvy users, including engineers, developers, researchers and students.

## Improper foreseeable use

The product is not suitable for industrial use or safety-relevant applications. Use of the product in medical devices or for aviation and space travel purposes is not permitted

## disposal

Do not discard with household waste! Your product is according to the European one Directive on waste electrical and electronic equipment to be disposed of in an environmentally friendly manner. The valuable raw materials contained therein can be recycled become. The application of this directive contributes to environmental and health protection. Use the collection point set up by your municipality to return and Recycling of old electrical and electronic devices. WEEE Reg. No.: DE 62624346

## electrostatic discharge

Attention: Electrostatic discharges can damage the product. Note: Ground yourself before touching the product, such as by wearing an anti-static wrist strap or touching a grounded metal surface.

## safety instructions

Although our product complies with the requirements of the RoHS Directive (2011/65/EU) and does not contain any hazardous substances in quantities above the permitted limits, residues may still be present. Observe the following safety instructions to avoid chemical hazards: Caution: Soldering can produce fumes that can be harmful to health. Note: Use a solder fume extractor or work in a well-ventilated area. If necessary, wear a respirator mask. Caution: Some people may be sensitive to certain materials or chemicals contained in the product. Note: If skin irritation or allergic reactions occur, stop use and, if necessary, consult a doctor. Caution: Keep the product out of the reach of children and pets to avoid accidental contact and swallowing of small parts. Note: Store the product in a safe, closed container when not in use. Attention: Avoid contact of the product with food and drinks. Note: Do not store or use the product near food to prevent contamination. Although our product complies with the requirements of the RoHS Directive (2011/65/EU) and does not contain any hazardous substances in quantities above the permitted limits, residues may still be present. Observe the following safety instructions to avoid chemical hazards: Caution: Soldering can produce fumes that can be harmful to health. Note: Use a solder fume extractor or work in a well-ventilated area. If necessary, wear a respirator mask. Caution: Some people may be sensitive to certain materials or chemicals contained in the product. Note: If skin irritation or allergic reactions occur, stop use and, if necessary,

consult a doctor. Caution: Keep the product out of the reach of children and pets to avoid accidental contact and swallowing of small parts. Note: Store the product in a safe, closed container when not in use. Attention: Avoid contact of the product with food and drinks. Note: Do not store or use the product near food to prevent contamination. The product contains sensitive electronic components and sharp edges. Improper handling or assembly can result in injury or damage. Observe the following safety instructions to avoid mechanical hazards: Attention: The product's circuit board and connectors may have sharp edges. Use caution to avoid cuts. Note: Wear appropriate protective gloves when handling and assembling the product. Caution: Avoid excessive pressure or mechanical stress on the board and components. Note: Only mount the product on stable and flat surfaces. Use appropriate spacers and housings to minimize mechanical stress. Attention: Make sure the product is securely fastened to prevent accidental slipping or falling. Note: Use appropriate support or secure mounting in enclosures or on mounting plates. Caution: Make sure all cable connections are connected securely and correctly to avoid strain and accidental unplugging. Note: Route cables so that they are not under tension and do not pose a tripping hazard. The product operates with electrical voltages and currents that, if used improperly, can result in electric shocks, short circuits or other hazards. Observe the following safety instructions to avoid electrical hazards: Attention: Use the product only with the specified voltages. Note: The performance limits of the product can be found in the associated data sheet Caution: Avoid short circuits between the connectors and components of the product Note: Make sure that no conductive objects touch or bridge the circuit board. Use insulated tools and pay attention to the arrangement of connections. Caution: Do not perform any work on the product when it is connected to a power source. Note: Disconnect the product from power before making any circuit changes or connecting or removing components. Caution: Do not exceed the specified current ratings for the product's inputs and outputs. Note: The performance limits of the product can be found in the technical specifications or in the data sheet Attention: Make sure that the power sources used are stable and correctly sized. Note: Only use tested and suitable power supplies to avoid voltage fluctuations and overloads. Attention: Maintain sufficient distance from live parts to avoid accidental contact. Note: Ensure that the cabling is arranged safely and clearly according to the voltage used. Caution: Use insulating housings or protective covers to protect the product from direct contact. Note: Place the product in a non-conductive case to avoid accidental touching and short circuits. The product and the components on it may become warm during operation. Improper handling or overloading the product can result in burns, damage or fire. Observe the following safety instructions to avoid thermal hazards: Caution: Make sure the product is used within recommended operating temperatures. Note: The recommended operating temperature range is typically between -40°C and +85°C. Check the specific information in the product data sheet. Attention: Do not place the product near external heat sources such as radiators or direct sunlight. Note: Ensure that the product is operated in a cool and well-ventilated area. Attention: Make sure the product is well ventilated to avoid overheating. Note: Use fans or heat sinks when operating the product in a closed enclosure or in an environment with limited air circulation. Attention: Mount the product on heat-resistant surfaces and in heat-resistant housings. Note: Use enclosure materials that can withstand high temperatures to avoid damage or fire hazard. Caution: Implement temperature monitoring when using an enclosure and, if necessary, protection mechanisms that shut down the product if it overheats. Note: Note: Use temperature sensors and appropriate software to monitor the temperature of the product and shut down the system if necessary. Caution: Avoid overloads that can cause excessive heating of components. Note: To prevent overheating, do not exceed the specified current and voltage limits. Caution: Short circuits can generate significant heat and cause fires. Note: Make sure that all connections are correct and secure and that no conductive objects can accidentally cause short circuits.

# Az-Delivery

The MAX6675 temperature sensor is an advanced thermocouple-to-digital converter with a built-in 12-bit ADC (analogue-to-digital converter) that performs cold junction compensation and digitises the signal from a type-k thermocouple. The function of the thermocouple is to detect temperature differences between two ends of the thermocouple wires.

The hot measuring point of the thermocouple can be read in a temperature range from 0°C to +1023.75°C. The temperature at the cold end ranges from -20°C to 85°C, which corresponds to the ambient temperature of the board on which the MAX6675 is mounted. While the temperature at the cold end of the thermocouple fluctuates, the MAX6675 detects and corrects the temperature changes by cold junction compensation. The MAX6675 converts the measured ambient temperature into voltage using a temperature sensing diode.

The MAX6675 measures the temperature by reading the voltage at the output of the thermocouple and the sensor diode. To achieve optimum performance of the MAX6675, the reference junction of the thermocouple and the MAX6675 must be at the same temperature. Therefore, avoid placing heat-generating devices and components in the vicinity of the MAX6675, as this can lead to reference junction errors.

# Az-Delivery

## Technical data:

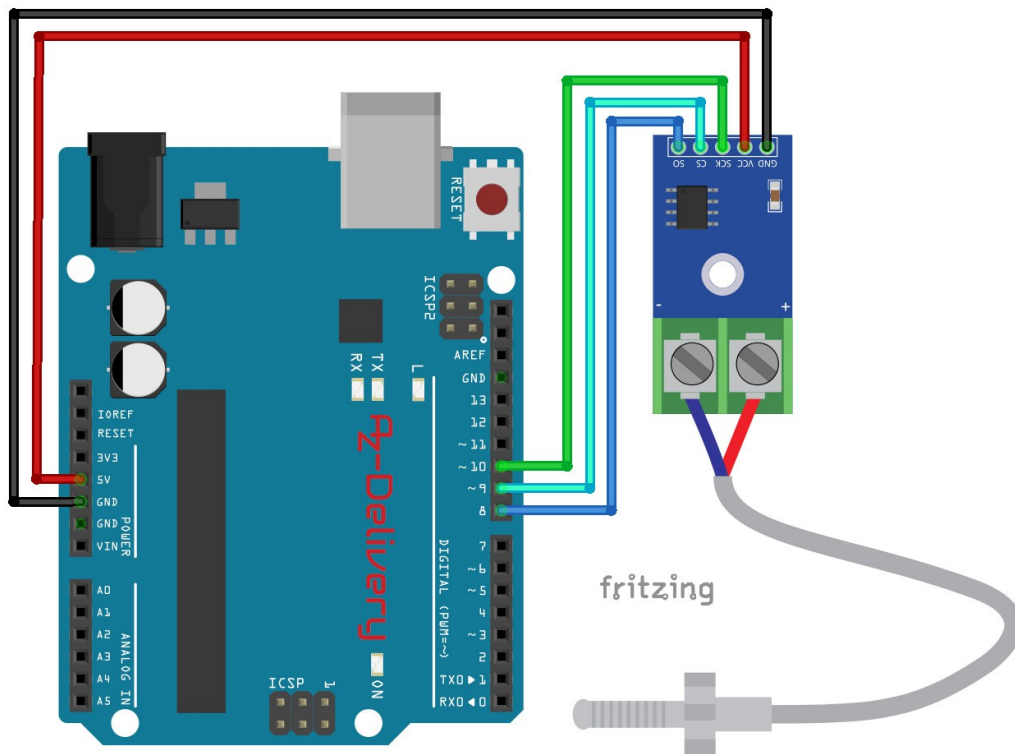
"	Operating voltage:	3.0V ~ 5.0V
"	Data output:	12 bit
"	Power supply:	1.5mA
"	Max. Output:	50mA
"	Output mode:	SPI digital signal
"	Temperature test range:	0°C to 1024°C
"	Temperature measurement accuracy:	+/- 1.5C
"	Temperature resolution:	0.25°C

The MAX6675 uses the SPI interface to communicate with microcontrollers. "SPI" stands for a synchronous serial peripheral communication interface and is used for communication over short distances, mainly in embedded systems. The SPI can be described as a synchronous serial interface. Synchronous means a complete transmission and reception of data within a certain clock cycle, and serial means that the data is sent serially, bit by bit.

# Az-Delivery

## Connecting the sensor to the Atmega328p

Connect everything as shown below:



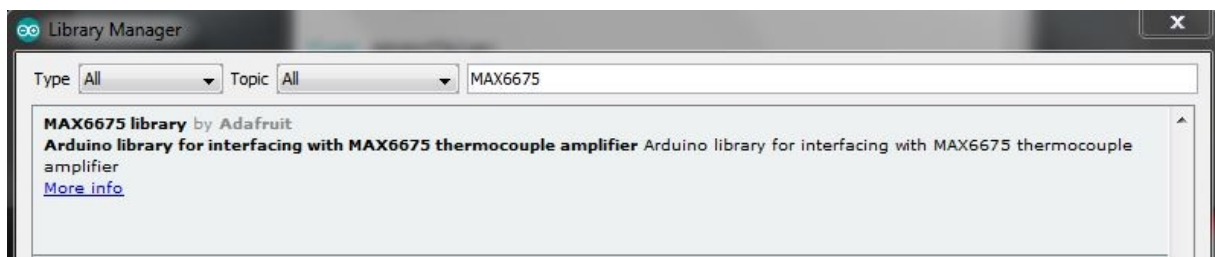
MAX6675 Module Pin			>Mc Pin	
VCC		>	5V	Red wire
GND		>	GND	Black wire
SCK	(CSK)	>	D10 Pin	Green wire
CS	(SS)	>	D9 Pin	Cyan wire
SO	(DO)	>	D8 Pin	Blue wire

Be sure to read the pin designations before you start connecting, as there are different versions of the module. We have included some alternative pin designations within the brackets.

# Az-Delivery

## Library

To use this module with the Atmega328p, we first need to download the required library. Go to *Tools > Manage Libraries*. In a new window that opens, enter "MAX6675" in the search field and install the "MAX6675" library from "Adafruit" as shown below:



Start the installation by clicking on the "Install" button that appears when you move the mouse over the search result.



## The Atmega328p sketch of the MAX6675 module

The sketch for this module is shown below:

```
#include "max6675.h"

int thermoDO = 8;
int thermoCS = 9;
int thermoCLK = 10;

MAX6675 thermocouple(thermoCLK, thermoCS, thermoDO);

void setup()
{
    Serial.begin(9600);
    Serial.println("MAX6675 test");
    delay(500);
}

void loop()
{
    Serial.print("C = ");
    Serial.println(thermocouple.readCelsius());
    Serial.print("F = ");
    Serial.println(thermocouple.readFahrenheit());

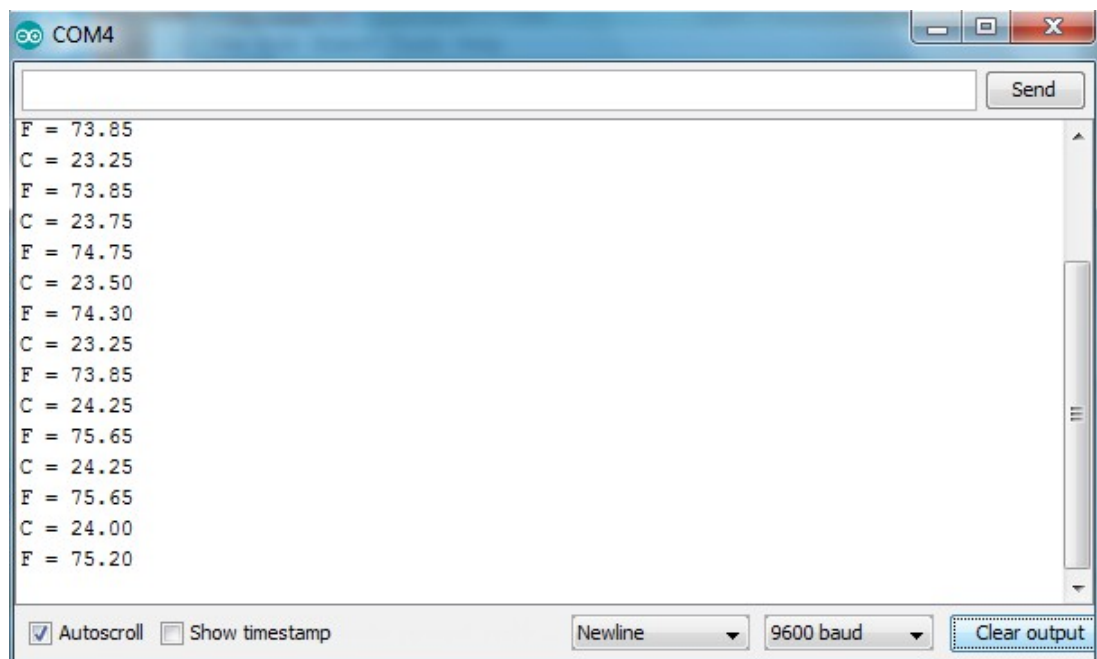
    delay(1000);
}
```



# Az-Delivery

The sketch starts by adding the "*MAX6675 Library*" that we have just installed. Next, we define which pins are to be used. In the `setup` function, we start the serial port with a "*baud rate of 9600*" and print the text "*MAX6675 test*" on the serial monitor. In the `loop` function, we carry out a simple readout test in which the current temperature is output in both Celsius and Fahrenheit.

To view results, open your serial monitor by going to *Tools > Serial Monitor*. The output should look like this:

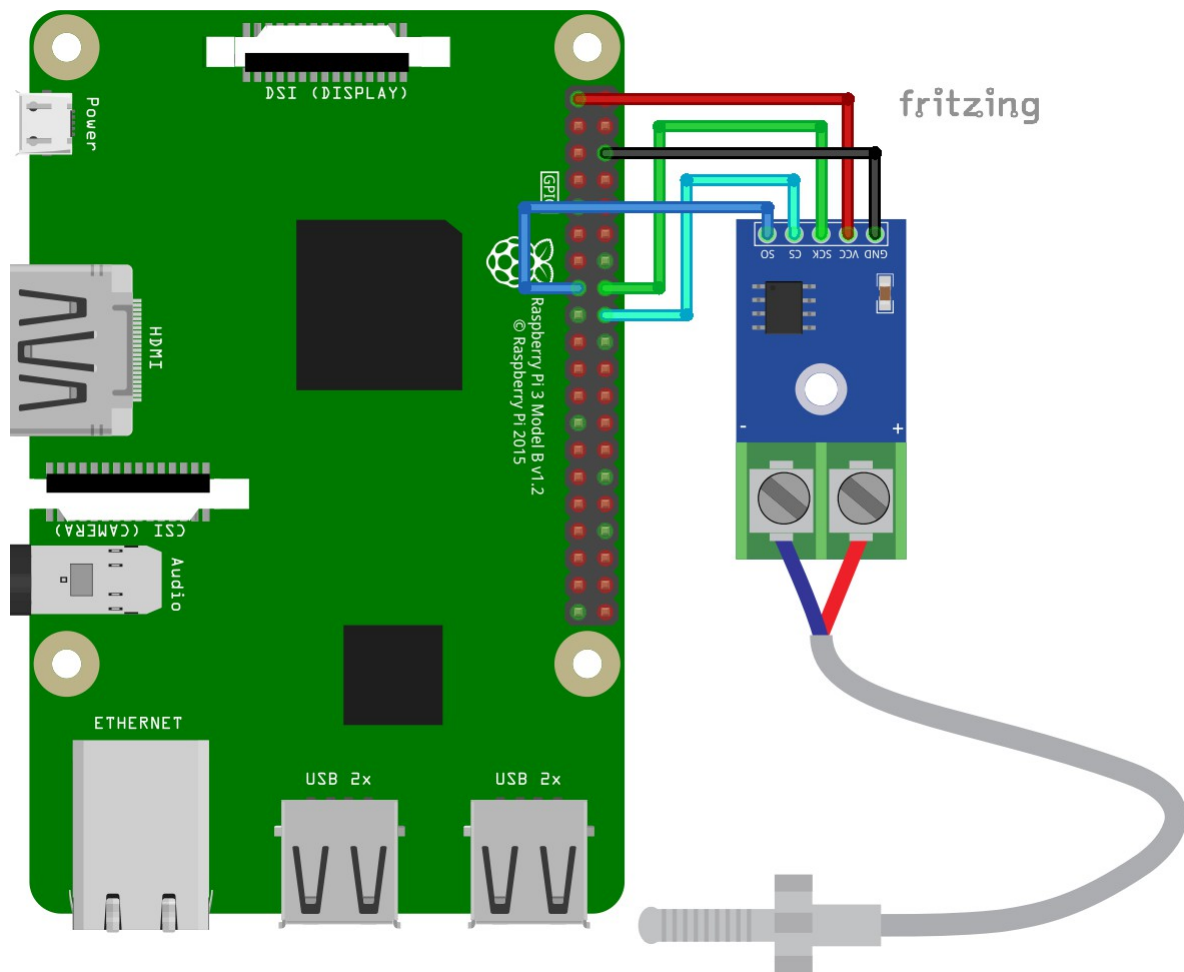


As you can see in the picture, the sensor measures the room temperature in Fahrenheit and Celsius. When the thermocouple is exposed to body heat (by placing it between your fingers), the temperature starts to rise.

# Az-Delivery

## MAX6675 sensor and the Raspberry Pi

Connect everything as shown below:



# Az-Delivery

MAX6675 module Pin		>Raspberry Pi Pin		
VCC		>	3.3V	Red wire
GND		>	GND	Black wire
SO	(DO)	>	GPIO22 [Pin 8]	Blue wire
SCK	(CSK)	>	GPIO23 [Pin 33]	Green wire
CS	(SS)	>	GPIO24 [Pin 10]	Cyan wire

Again, we have included some alternative pin names in brackets, as there are different versions of the module. Once you have connected everything, we can continue with the programming in Python code.

# Az-Delivery

## The Python code

For clarity, we will write two scripts. The class script is shown below:

```
import RPi.GPIO as GPIO
import time

class MAX6675(object):

    def init (_self, cs_pin, clock_pin, data_pin, units = "c", board = GPIO.BCM): #
        Initialise Soft (Bitbang) SPI bus
        # Parameters:
        # - cs_pin:      ChipSelect (CS) / Slave Select (SS) pin (Any GPIO)
        # - clock_pin: Clock (SCLK / SCK) pin (Any GPIO)
        # - data_pin: Data input (SO / MOSI) pin (Any GPIO)
        # - units: (optional) unit of measurement to return. ("c" (default) | "k" | "f")
        # - board: (optional) pin numbering method as per RPi.GPIO library (GPIO.BCM
        #           (default) | GPIO.BOARD)
        self.cs_pin = cs_pin
        self.clock_pin = clock_pin
        self.data_pin = data_pin
        self.units = units
        self.data = None
        self.board = board

        # Initialise needed GPIO
        GPIO.setmode(self.board)
        GPIO.setwarnings(False)
        GPIO.setup(self.cs_pin, GPIO.OUT)
        GPIO.setup(self.clock_pin, GPIO.OUT)
        GPIO.setup(self.data_pin, GPIO.IN) #

        # Pull chip select high to make chip inactive
        GPIO.output(self.cs_pin, GPIO.HIGH)
```

# Az-Delivery

```
def get(self):
    # Reads SPI bus and returns current value of thermocouple.
    self.read()
    self.checkErrors()
    return getattr(self, "to_" + self.units)(self.data_to_tc_temperature())

def read(self):
    # Reads 16 bits of the SPI bus & stores as an integer in self.data.
    bytesin = 0
    # Select the chip
    GPIO.output(self.cs_pin, GPIO.LOW)
    # Read in 16 bits
    for i in range(16):
        GPIO.output(self.clock_pin, GPIO.LOW)
        time.sleep(0.001)
        bytesin = bytesin << 1
        if (GPIO.input(self.data_pin)):
            bytesin = bytesin | 1
        GPIO.output(self.clock_pin, GPIO.HIGH)
    time.sleep(0.001)
    # Unselect the chip
    GPIO.output(self.cs_pin, GPIO.HIGH)
    # Save data
    self.data = bytesin

def checkErrors(self, data_16 = None):
    # Checks errors on bit D2
    if data_16 is None:
        data_16 = self.data
    noConnection = (data_16 & 0x4) != 0 # tc input bit, D2 if
    noConnection:
        raise MAX6675Error("No Connection") # open thermocouple

def data_to_tc_temperature(self, data_16 = None):
    # Takes an integer and returns a thermocouple temperature in celsius.
    if data_16 is None:
        data_16 = self.data
    # Remove bits D0-3
    tc_data = ((data_16 >> 3) & 0xFFFF)
    # 12-bit resolution
    return (tc_data * 0.25)
```

# Az-Delivery

```
def to_c(self, celsius):
    # Celsius passthrough for generic to_* method.
    return celsius

def to_k(self, celsius):
    # Convert celsius to kelvin.
    return celsius + 273.15

def to_f(self, celsius):
    # Convert celsius to fahrenheit.
    return celsius * 9.0/5.0 + 32

def cleanup(self):
    # Selective GPIO cleanup
    GPIO.setup(self.cs_pin, GPIO.IN)
    GPIO.setup(self.clock_pin, GPIO.IN)
    GPIO.cleanup()

class MAX6675Error(Exception):
    def init (_self, value):
        self.value = value
    def str (self):
        return repr(self.value)
```

# Az-Delivery

```
if name_____ == " main _":

    # default example
    cs_pin = 24
    clock_pin = 23
    data_pin = 22
    units = "c"
    thermocouple = MAX6675(cs_pin, clock_pin, data_pin, units)
    time.sleep(2)
    try:
        while True:
            try:
                tc = thermocouple.get()

                except MAX6675Error as e:
                    tc = "Error: "+ e.value
                    running = False
                    print("tc: {}".format(tc))

            time.sleep(1)

    except KeyboardInterrupt:
        print("Script End!")
        thermocouple.cleanup()
```

Note that a large part of the code has been taken from this website:

<https://github.com/Tuckie/max31855/blob/master/max31855.py#L11> Save this script under the name "*mx6675class.py*".

# Az-Delivery

The main script is shown below:

```
import time
from max6675class import MAX6675, MAX6675Error

cs_pin = 24
clock_pin = 23
data_pin = 22
units = "c"
degree_sign = u'\xb0'

print("[press ctrl+c to end the script]")
try: # Main program loop
    while True:
        thermocouple = MAX6675(cs_pin, clock_pin, data_pin, units)
        time.sleep(2)
        print("Temperature\t{} {}C".format(thermocouple.get(), degree_sign))

except KeyboardInterrupt:
    print("Script end!")
    thermocouple.cleanup()
```

Save this script as *"max6675read.py"* in the same directory as the first script. Now we will explain the code. In the first script, we created the classes *"MAX6675"* and *"MAX6675Error"* to be used in the second script. The *MAX6675 class* does the reading, while *MAX6675Error* finds errors. In the main script we have imported the created classes, defined which pins to use and which units to display. In the main programme loop, the programme reads and writes the temperature every two seconds. And finally, we have set up *KeyboardInterrupt*.

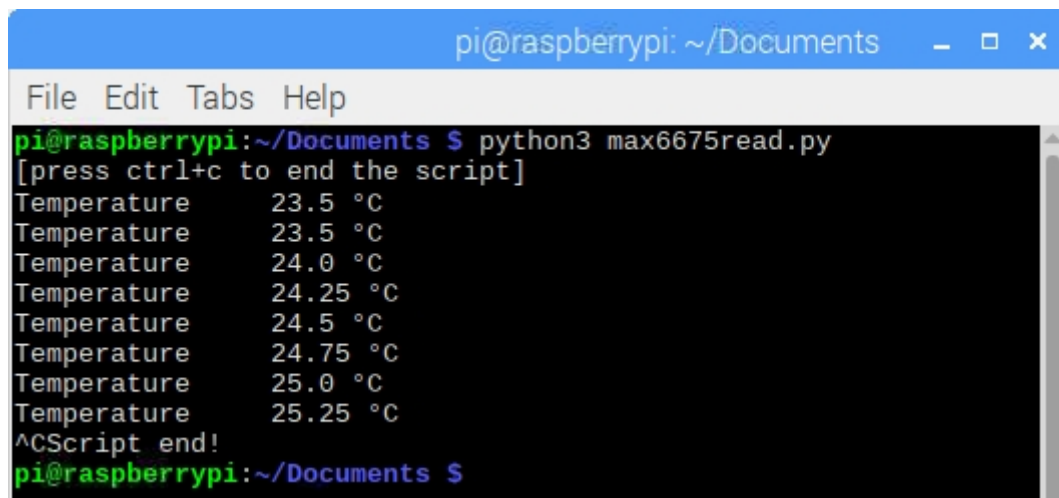


# Az-Delivery

After all steps have been completed, we start the script by executing the following command:

```
python3 max6675read.py
```

The output should look like this:



```
pi@raspberrypi: ~/Documents
File Edit Tabs Help
pi@raspberrypi:~/Documents $ python3 max6675read.py
[press ctrl+c to end the script]
Temperature      23.5 °C
Temperature      23.5 °C
Temperature      24.0 °C
Temperature      24.25 °C
Temperature      24.5 °C
Temperature      24.75 °C
Temperature      25.0 °C
Temperature      25.25 °C
^CScript end!
pi@raspberrypi:~/Documents $
```

As you can see in the picture, the sensor measures the room temperature in Celsius. When the thermocouple was exposed to body heat (between the fingers), the temperature began to rise. Pressing CTRL+C stops the process.

**You have done it. You can now use our module  
for your projects.**



Now it's your turn! Develop your own projects and smart home installations. We will show you how to do this in an uncomplicated and understandable way on our blog. There we offer you sample scripts and tutorials with interesting small projects to get you started quickly in the world of microelectronics. The Internet also offers you countless opportunities to learn more about microelectronics.

**If you are looking for more high quality microelectronics and accessories, AZ-Delivery Vertriebs GmbH is the right place for you. We offer you numerous application examples, detailed installation instructions, e-books, libraries and of course the support of our technical experts.**

<https://az-delivery.de>

Have fun!

Imprint

<https://az-delivery.de/pages/about-us>