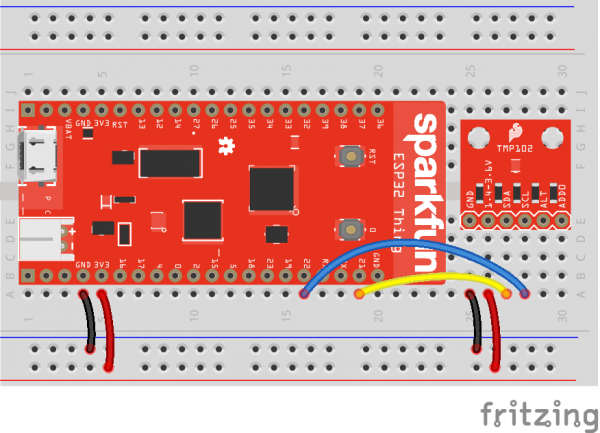
Experiment 4: I2C

Inter-Integrated Circuit (I2C) is a communication protocol designed to allow one "master" chip to communicate with several "device" chips on the same bus. Many modern sensors rely on I2C for communication, as it is a relatively easy protocol to implement and requires only 2 signal lines. If you would like to learn more about how I2C works, see [this tutorial](https://learn.sparkfun.com/tutorials/i2c).

In this experiment, we are going to connect a [TMP102 temperature sensor](https://www.sparkfun.com/products/13314) to the ESP32 Thing and read the ambient temperature.

Hardware Connections

Connect the TMP102 to the ESP32 as follows:

[](https://cdn.sparkfun.com/assets/learn_tutorials/6/5/9/esp32_micropython_03_bb.png)

COPY CODE**import** machine

**import** sys

**import** utime

# Pin definitions

repl\_button **=** machine.Pin(0, machine.Pin.IN, machine.Pin.PULL\_UP)

repl\_led **=** machine.Pin(5, machine.Pin.OUT)

sda\_pin **=** machine.Pin(21)

scl\_pin **=** machine.Pin(22)

# Create an I2C object out of our SDA and SCL pin objects

i2c **=** machine.I2C(sda**=**sda\_pin, scl**=**scl\_pin)

# TMP102 address on the I2C bus

tmp102\_addr **=** 0x48

# TMP102 register addresses

reg\_temp **=** 0x00

reg\_config **=** 0x01

# Functions

# Calculate the 2's complement of a number

**def** **twos\_comp**(val, bits):

**if** (val **&** (1 **<<** (bits **-** 1))) **!=** 0:

val **=** val **-** (1 **<<** bits)

**return** val

# Read temperature registers and calculate Celsius

**def** **read\_temp**():

# Read temperature registers

val **=** i2c.readfrom\_mem(tmp102\_addr, reg\_temp, 2)

temp\_c **=** (val[0] **<<** 4) **|** (val[1] **>>** 5)

# Convert to 2s complement (temperatures can be negative)

temp\_c **=** twos\_comp(temp\_c, 12)

# Convert registers value to temperature (C)

temp\_c **=** temp\_c **\*** 0.0625

**return** temp\_c

# Initialize communications with the TMP102

**def** **init**():

# Read CONFIG register (2 bytes) and convert to integer list

val **=** i2c.readfrom\_mem(tmp102\_addr, reg\_config, 2)

val **=** list(val)

# Set to 4 Hz sampling (CR1, CR0 = 0b10)

val[1] **=** val[1] **&** 0b00111111

val[1] **=** val[1] **|** (0b10 **<<** 6)

# Write 4 Hz sampling back to CONFIG

i2c.writeto\_mem(tmp102\_addr, reg\_config, bytearray(val))

###############################################################################

# Main script

# Print out temperature every second

**while** True:

# If button 0 is pressed, drop to REPL

**if** repl\_button.value() **==** 0:

print("Dropping to REPL")

repl\_led.value(1)

sys.**exit**()

# Read temperature and print it to the console

temperature **=** read\_temp()

print(round(temperature, 2), "C")

utime.sleep(1)

Save the file with a name such as *i2c.py*. In a command terminal, navigate to the directory with your *i2c.py* file. Hold down button 0 until you see the blue LED turn on to let you know that you have entered the REPL. In the command terminal, enter the following commands (change <PORT> to your port name/location):

COPY CODEcp i2c.py main.py

ampy --port **<**PORT**>** put main.py

Open a serial connection to the ESP32, and you should see temperature value (in Celsius) being reported once per second.

Try breathing warm air onto the TMP102 sensor to see if you can change the temperature values.

[](https://cdn.sparkfun.com/assets/learn_tutorials/6/5/9/screen_09.png)

Code to Note

This code is much longer than our previous examples. This is due to us needing to send configuration information to the TMP102, read temperature, and then perform some calculations on the raw data to get a human-readable temperature value.

Every time we want to communicate with the TMP102, we send out its address (0x48) on the bus. From there, we usually send the memory location (address) of the register that we intend to read from or write to on the TMP102. For most I2C devices, a *register* is a location in the device's memory that stores 1 byte (8 bits) of data. Some registers control the function of the device (for example, the CONFIG register in the TMP102). Other registers hold sensor data readings (such as the TEMP register in the TMP102) that we must read from.

To communicate on an I2C bus, we create a machine.I2C object, passing it the pins we intend to use for SDA and SCL:

COPY CODEi2c **=** machine.I2C(sda**=**sda\_pin, scl**=**scl\_pin)

We use the following command to read 2 bytes from the temperature register in the TMP102:

COPY CODEval **=** i2c.readfrom\_mem(tmp102\_addr, reg\_temp, 2)

These values are stored as a list [x, y] in the val variable. By looking at the TMP102 datasheet, we see that temperature is 12 bits. When we read the two bytes that contain this reading, we need to remove the last 4 bits from the second byte. We also move the first byte over 4 bits:

COPY CODEtemp\_c **=** (val[0] **<<** 4) **|** (val[1] **>>** 5)

In order to display negative numbers for the temperature, values from the TMP102 come in the form of Two’s Complement. In this, the first bit of the 12-bit number determines if the value is positive or negative (0 for positive, 1 for negative). See [this article](https://www.cs.cornell.edu/~tomf/notes/cps104/twoscomp.html) to learn more about Two’s Complement.

To convert a Two’s Complement number to a negative number in Python, we check to see if the first bit is 0 or 1. If it is 0, then we just use the number as is (it’s positive!). If it’s a 1, we subtract the max negative number of the Two’s Complement (212=4096 in this case) from our number.

COPY CODE**if** (val **&** (1 **<<** (bits **-** 1))) **!=** 0:

val **=** val **-** (1 **<<** bits)

Finally, we multiply this two's complement number by 0.0625 to convert the raw value into a Celsius value. Refer out Table 5 in the [TMP102 datasheet](https://www.sparkfun.com/datasheets/Sensors/Temperature/tmp102.pdf) to see how the raw-to-Celsius calculation is performed.