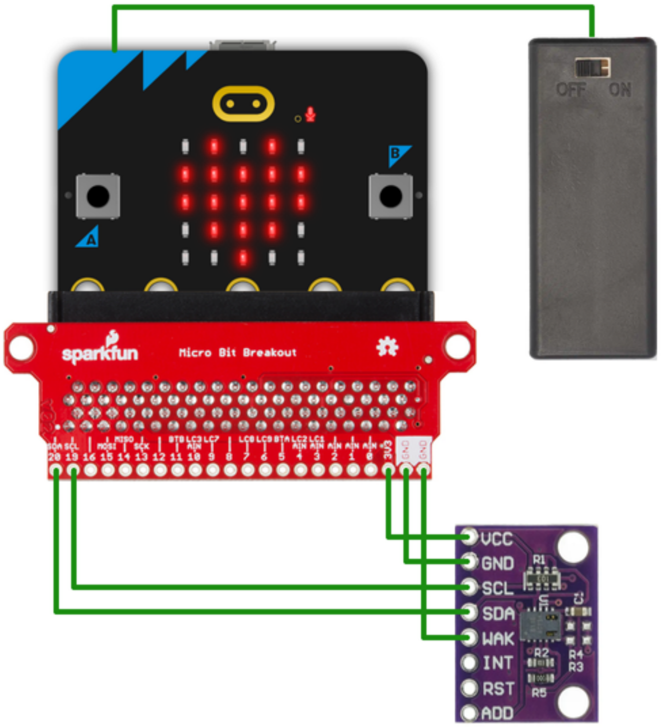
05.Technical documentation (2023-2024-S019-S020-S021)

01. Micro:Bit and CO2 sensor

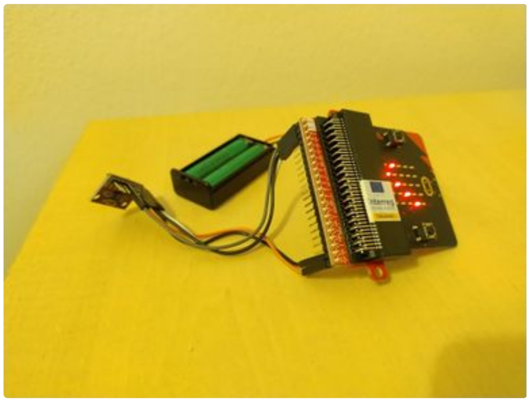
Scheme

5 cables connecting the Micro:Bit breakout pins with the CO2 sensor according to the diagram:

```
VCC -> 3V3
GND -> GND
SCL -> SCL (19)
SDA -> SDA (20)
WAK -> GND
```



Solution



Assembling the Micro:Bit and CCS811

1. Connect the Co2 Sensor with the Micro:Bit breakout pins as illustrated in the diagram. Make sure to solder the pins properly to the sensor.
2. Connect the Microbit to the PC with an USB cable. The Micro:Bit will turn on.
3. Open micro:bit Python Editor and paste the source code (available below) to the main.py file of your project.
4. Connect your microbit with the “Send to Micro:Bit“ button.
5. If everyting works, disconnect the USB cable and connect your batteries. At this point, you can assemble the solution in the case.

Comments

- be sure to connect the cables properly
- sensor only measures the CO2 concentration against a baseline, which calibrates itself automatically - that is why is it best to let the sensor calibrate for 20 minutes (possibly in a room with low CO2?)
- sensor has an option to be calibrated manually, this is not a part of our solution, please read the datasheet (Datasheet) to find more

Code

```
# imports
from microbit import i2c, display, Image, sleep
import music

# images
img_ok = Image('00000:00009:00090:90900:09000')
img_warning = Image('00900:00900:00900:00000:00900')
img_danger = Image('90909:90909:90909:00000:90909')
img_wait = Image('09990:90909:90909:90099:09990')

# format two byte register data
def format(arr):
    return ((arr[0] << 8) | arr[1])

# check if sensor data is ready
def data_ready():
    i2c.write(90, bytearray([0x00]))
    return (i2c.read(90, 1)[0] >> 3) & 0x01

# reads the co2, voc and baseline data
def read_data():
    i2c.write(90, bytearray([0x02]))
    register1 = i2c.read(90, 4)
    i2c.write(90, bytearray([0x11]))
    register2 = i2c.read(90, 2)
    return (
        format(register1[0:2]),
        format(register1[2:4]),
        format(register2))

# initiate the sensor
# help from https://github.com/Notthemarsian/CCS811/blob/master/CCS811.py
if 90 not in i2c.scan():
    raise ValueError('CCS811 not found.')
i2c.write(90, bytearray([0x20]))
if (i2c.read(90, 1)[0] != 0x81):
    raise ValueError('Wrong Hardware ID.')
i2c.write(90, bytearray([0x00]))
if not (i2c.read(90, 1)[0] >> 4) & 0x01:
    raise ValueError('Application not valid.')
i2c.write(90, bytearray([0xF4]))
i2c.write(90, bytearray(b'\x01\x18'))

# initial image
display.show(img_wait)
sleep(5000)
display.clear()

# endless loop
while True:
    if data_ready():
        display.clear()
        co2, _, _ = read_data()

        # show co2 particles
        display.scroll(str(co2))
        display.clear()

        # show image based on the co2 value
        if co2 < 1200:
            display.show(img_ok)
        elif co2 < 1500:
            display.show(img_warning)
        else:
            display.show(img_danger)
            music.play(music.BA_DING)

        # wait
        sleep(18000)
        display.show(img_wait)
    sleep(2000)
```

02. 3D Model

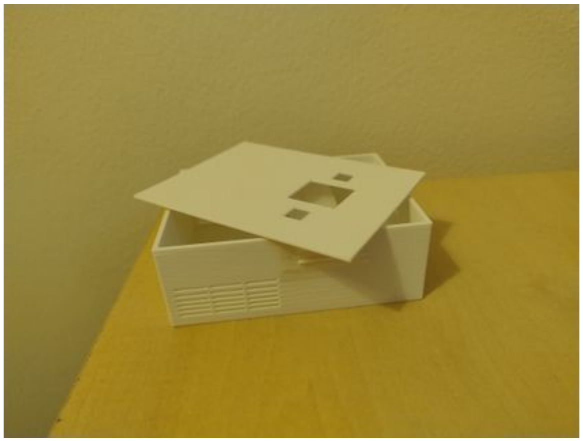
Files

3D design Sensor frame | Tinkercad

Sensor_frame_v2.3mf
06 Dec 2023, 12:28 pm

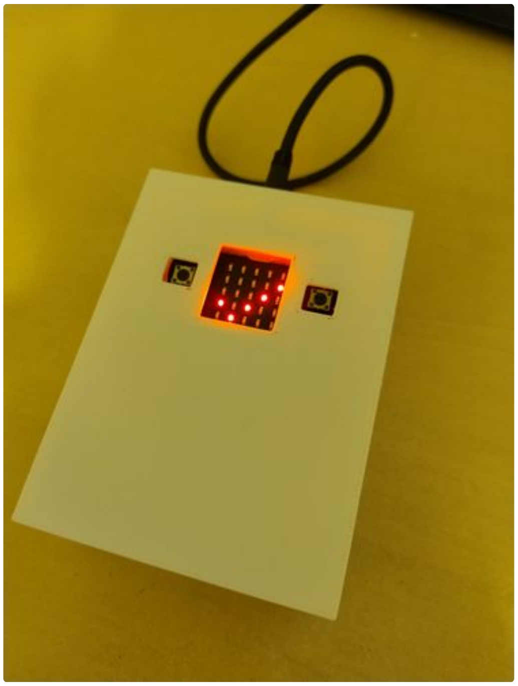
Sensor_frame_v2.stl
06 Dec 2023, 12:28 pm

Final solution

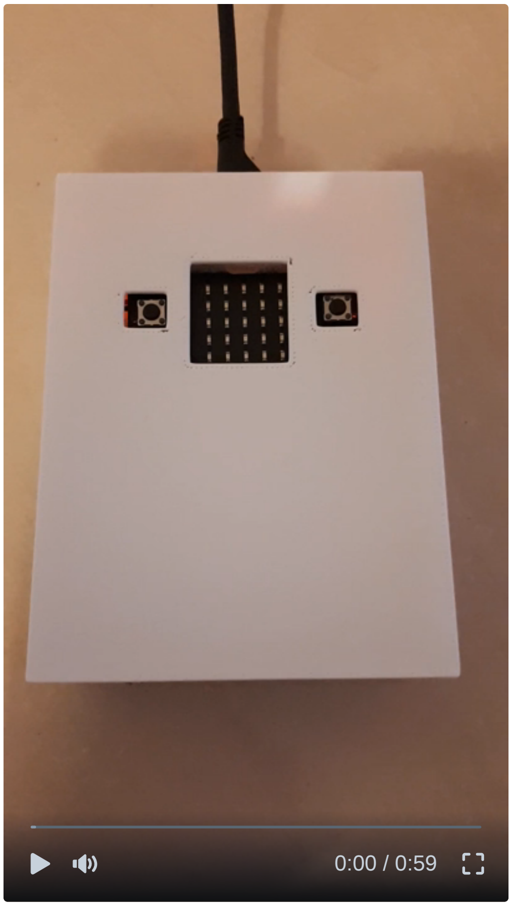


03. Showcase

Working solution in the case:



Demo of possible CO2 levels:



Demo of the working sensor (opening a window to lower the CO2 concentration):