Custom serial sensor

Jean-claude.feltes@education.lu

My config.yaml uses a custom serial sensor located in custom_components/serial_jc1
It was elaborated with the help of ChatGPT (but not without human intelligence!).

Why did I (re-)write this?
Because in the original there was no error handling for

```
line = line.decode("utf-8")
```

This caused an error that was difficult to trace. I only found it when I had a look at the component source code. The sensor stopped sometimes when it got values that it could not decode.
This could happen when the connection was interrupted and then reconnected.

Custom sensor folder

The folder homeassistant/custom_components/serial_jc1 contains 3 files:

__init__.py
manifest.json
sensor.py

__init__.py

```
from .sensor import CustomSerialSensor
```

manifest.json

```
{
  "domain": "serial_jc1",
  "name": "Serial JC",
  "documentation": "https://www.home-assistant.io/integrations/serial",
  "iot_class": "local_polling",
  "requirements": ["pyserial-asyncio==0.6"],
  "version": "1.0.0",
  "config_flow": true
}
```

version is important. If no version info, the component is not loaded!

sensor.py

```
import asyncio
import logging

import serial_asyncio
from serial import SerialException
from homeassistant.components.sensor import PLATFORM_SCHEMA, SensorEntity
```

```
from homeassistant.const import CONF_NAME, CONF_VALUE_TEMPLATE
from homeassistant.core import HomeAssistant
import homeassistant.helpers.config_validation as cv
from homeassistant.helpers.entity_platform import AddEntitiesCallback
from homeassistant.helpers.typing import ConfigType, DiscoveryInfoType
import voluptuous as vol

_LOGGER = logging.getLogger(__name__)

CONF_SERIAL_PORT = "serial_port"
CONF_BAUDRATE = "baudrate"

DEFAULT_NAME = "Serial JC1 Sensor"
DEFAULT_BAUDRATE = 9600

PLATFORM_SCHEMA = PLATFORM_SCHEMA.extend(
    {
        vol.Required(CONF_SERIAL_PORT): cv.string,
        vol.Optional(CONF_BAUDRATE, default=DEFAULT_BAUDRATE): cv.positive_int,
        vol.Optional(CONF_NAME, default=DEFAULT_NAME): cv.string,
        vol.Optional(CONF_VALUE_TEMPLATE): cv.template,
    }
)


async def async_setup_platform(
    hass: HomeAssistant,
    config: ConfigType,
    async_add_entities: AddEntitiesCallback,
    discovery_info: DiscoveryInfoType | None = None,
) -> None:
    """Set up the Serial sensor platform."""
    name = config.get(CONF_NAME)
    port = config.get(CONF_SERIAL_PORT)
    baudrate = config.get(CONF_BAUDRATE)

    if (value_template := config.get(CONF_VALUE_TEMPLATE)) is not None:
        value_template.hass = hass

    sensor = CustomSerialSensor(name, port, baudrate, value_template)
    async_add_entities([sensor], True)


class CustomSerialSensor(SensorEntity):
    """Representation of a Serial sensor."""

    _attr_should_poll = False

    def __init__(
        self,
        name,
        port,
        baudrate,
        value_template,
    ):
        """Initialize the Serial sensor."""
        self._name = name
        self._state = None
        self._port = port
        self._baudrate = baudrate
        self._serial_loop_task = None
        self._template = value_template

    async def async_added_to_hass(self) -> None:
        """Handle when an entity is about to be added to Home Assistant."""
        self._serial_loop_task = self.hass.loop.create_task(self.serial_read())

    async def serial_read(self):
        """Read the data from the port."""
        logged_error = False
        while True:
            try:
                reader, _ = await serial_asyncio.open_serial_connection(
                    url=self._port,
                    baudrate=self._baudrate,
                )
```

```
            except SerialException as exc:
                if not logged_error:
                    _LOGGER.exception(
                        "Unable to connect to the serial device %s: %s. Will retry",
                        self._port,
                        exc,
                    )
                    logged_error = True
                await self._handle_error()
            else:
                _LOGGER.info("Serial device %s connected", self._port)
                while True:
                    try:
                        line = await reader.readline()
                        line = line.decode("utf-8").strip()
                    except SerialException as exc:
                        _LOGGER.exception(
                            "Error while reading serial device %s: %s", self._port, exc
                        )
                        await self._handle_error()
                        break
                    else:
                        if self._template is not None:
                            line = self._template.async_render_with_possible_json_value(
                                line
                            )

                        _LOGGER.debug("Received: %s", line)
                        self._state = line
                        self.async_write_ha_state()

    async def _handle_error(self):
        """Handle error for serial connection."""
        self._state = None
        self.async_write_ha_state()
        await asyncio.sleep(5)

    @property
    def name(self):
        """Return the name of the sensor."""
        return self._name

    @property
    def native_value(self):
        """Return the state of the sensor."""
        return self._state
```

Usage

In my configuration.yaml the sensor is used like this:

```
sensor:

  - platform: serial_jc1
    serial_port: /dev/ttyUSB0
    baudrate: 9600
    name: serial_data
    value_template: "{{ value[:250] }}"
```

Debugging

It is useful to have a look at the files home-assistant.log and home-assistant.log.1 with the file editor. And I confess, I had to do it many times before my component was ready.