GY-521 am ESP32 D1 R32 – Verbinden und Daten auslesen mit MicroPython

Verbinde GY-521 mit ESP32 D1 R32

VVC	rot	3.3V
GND	blau	GND
SCL	gelb	SCL / GPIO22
SDA	grün	SDA / GPIO21

```
>>> from machine import I2C
>>> i2c = I2C(sda=Pin(21), scl=Pin(22), freq=10000)
 Warning: I2C(-1, ...) is deprecated, use SoftI2C(...)
 instead
```

Kommandozeile

```
>>> from machine import Pin, SoftI2C
>>> i2c = SoftI2C(sda=Pin(21), scl=Pin(22), freq=10000)
>>> i2c.scan()
[104]
>>> hex(104)
'0x68'
>>> data = i2c.readfrom_mem( 0x68, 0x3B , 14)
>>> data
b'\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00'
>>> for b in data:
        print(b, end=" ")

 0 0 0 0 0 0 0 0 0 0 0 0 0 0
>>> data[0]
0
```

```
>>> i2c.start()
>>> i2c.writeto(0x68, bytearray([107, 0]))
2
>>> i2c.stop()
>>> data = i2c.readfrom_mem(0x68,0x3B,14)
>>> data
b'B\xb4\xfe\xb4\xfa@\xf1`\xfe\xbd\x01i\xffZ'
>>> for b in data:
        print(b, end=" ")

 66 180 254 180 250 64 241 96 254 189 1 105 255 90
```

```
>>> dir(i2c)
['__class__', 'readinto', 'start', 'stop', 'write', '
init', 'readfrom', 'readfrom_into', 'readfrom_mem', '
readfrom_mem_into', 'scan', 'writeto', 'writeto_mem',
'writevto']
```

Die Angaben für Adressen der Register finden wir in der MPU-6000-Register-Map1.pdf, aber nicht im DataSheet.

4.28 Register 107 – Power Management 1
PWR_MGMT_1

Type: Read/Write

Register (Hex)	Register (Decimal)	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
6B	107	DEVICE_RESET	SLEEP	CYCLE	-	TEMP_DIS	CLKSEL[2:0]		

Description:

This register allows the user to configure the power mode and clock source. It also provides a bit for resetting the entire device, and a bit for disabling the temperature sensor.

By setting SLEEP to 1 the MPU-60X0 can be put into low power sleep mode. When CYCLE is set to

4.17 Registers 59 to 64 – Accelerometer Measurements
ACCEL_XOUT_H, ACCEL_XOUT_L, ACCEL_YOUT_H, ACCEL_YOUT_L, ACCEL_ZOUT_L

Type: Read Only

Register (Hex)	Register (Decimal)	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
3B	59	ACCEL_XOUT[15:8]							
3C	60	ACCEL_XOUT[7:0]							
3D	61	ACCEL_YOUT[15:8]							
3E	62	ACCEL_YOUT[7:0]							
3F	63	ACCEL_ZOUT[15:8]							
40	64	ACCEL_ZOUT[7:0]							

Description:

These registers store the most recent accelerometer measureme

Accelerometer measurements are written to these registers a

4.19 Registers 67 to 72 – Gyroscope Measurements
GYRO_XOUT_H, GYRO_XOUT_L, GYRO_YOUT_H, GYRO_YOUT_L, GYRO_ZOUT_H, and GYRO_ZOUT_L

Type: Read Only

Register (Hex)	Register (Decimal)	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
43	67	GYRO_XOUT[15:8]							
44	68	GYRO_XOUT[7:0]							
45	69	GYRO_YOUT[15:8]							
46	70	GYRO_YOUT[7:0]							
47	71	GYRO_ZOUT[15:8]							
48	72	GYRO_ZOUT[7:0]							

scope measurements.

ese registers at the Sample Rate as defined in Register

4.18 Registers 65 and 66 – Temperature Measurement
TEMP_OUT_H and TEMP_OUT_L

Type: Read Only

Register (Hex)	Register (Decimal)	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
41	65	TEMP_OUT[15:8]							
42	66	TEMP_OUT[7:0]							

Description:

These registers store the most recent temperature sensor measurement.

Temperature measurements are written to these registers at the Sample Rate as defined in Register

https://joy-it.net/files/files/Produkte/SEN-MPU6050/SEN-MPU6050_Datenblatt_2021-11-04.pdf

aus https://microcontrollerslab.com/micropython-mpu-6050-esp32-esp8266/

- https://raw.githubusercontent.com/adamjezek98/MPU6050-ESP8266-MicroPython/master/mpu6050.py

```
>>> import mpu6050_1
>>> dir(mpu6050_1)
['__class__', '__name__', '__file__', 'machine', 'accel']
>>> dir(mpu6050_1.accel)
['__class__', '__init__', '__module__', '__name__', '__qualname__', '__bases__', '__dict__', 'get_raw_va
lues', 'get_ints', 'bytes_toint', 'get_values', 'val_test']
```

```
b'B\x8c\xfe\x9c\xf6\x90\xf0`\xfe\xbc\x01N\xffW'
{'GyZ': -169, 'GyY': 334, 'GyX': -324, 'Tmp': 24.76529, 'AcZ': -2416, 'AcY': -356, 'AcX': 17036}
  17036    -356   -2416
   -324     334    -169
```

https://microdigisoft.com/mpu6050-with-raspberry-pi-pico-using-micropython/

gyro.py angepasst uv.

https://mascil.ph-freiburg.de/aufgabensammlung/experimente-mit-dem-smartphone/gruppe-2-experimente-mit-dem-beschleunigungssensor#Anleitung_09

https://www.imw.tu-clausthal.de/fileadmin/Forschung/InstMitt/2013/107-118_Hofmann_Numerische.pdf

https://www.cik-solutions.com/content/files/schockmessung-grundlagen__copyright_msr_electronics_gmbh_180322_sec.pdf

https://www.cosinuss.com/de/2021/02/08/messung-von-vital-beschleunigungsparametern-beim-schwimmen/#Beschleunigungssensor

https://matplotlib.org/2.0.2/mpl_toolkits/mplot3d/tutorial.html

https://forum.arduino.cc/t/mpu6050-aus-dem-beschleunigungssensor-die-geschwindigkeit-berechnen/611507

https://www.htwg-konstanz.de/fileadmin/pub/studium/zsb/Schnupperstudium/2021-Laboranleitung-Beschleunigungssensor.pdf

https://www.physikalische-schulexperimente.de/physo/Beschleunigungssensor_im_Smartphone

https://www.elektronikpraxis.de/bewegungserkennung-mit-beschleunigungssensor-a-231973/?p=3

https://www.youtube.com/watch?v=zDGvcHMSPm8

In Pyton will ich die Daten seriell einlesen

s

```
Serial<id=0x17899ae5300, open=True>(port='COM8', baudrate=9600, bytesize=8, parity='N', stopbits=1, timeout=None,
xonxoff=False, rtscts=False, dsrdtr=False)
```

```
dir(s)
```

```
['BAUDRATES', 'BYTESIZES', 'PARITIES', 'STOPBITS', '_GetCommModemStatus', '_SAVED_SETTINGS', '__abstractmethods__',
'__class__', '__del__', '__delattr__', '__dict__', '__dir__', '__doc__', '__enter__', '__eq__', '__exit__',
'__format__', '__ge__', '__getattribute__', '__gt__', '__hash__', '__init__', '__init_subclass__', '__iter__',
'__le__', '__lt__', '__module__', '__ne__', '__new__', '__next__', '__reduce__', '__reduce_ex__', '__repr__',
'__setattr__', '__sizeof__', '__str__', '__subclasshook__', '_abc_impl', '_baudrate', '_break_state', '_bytesize',
'_cancel_overlapped_io', '_checkClosed', '_checkReadable', '_checkSeekable', '_checkWritable', '_close', '_dsrdtr',
'_dtr_state', '_exclusive', '_inter_byte_timeout', '_orgTimeouts', '_overlapped_read', '_overlapped_write',
'_parity', '_port', '_port_handle', '_reconfigure_port', '_rs485_mode', '_rts_state', '_rtscts', '_stopbits',
'_timeout', '_update_break_state', '_update_dtr_state', '_update_rts_state', '_write_timeout', '_xonxoff',
'applySettingsDict', 'apply_settings', 'baudrate', 'break_condition', 'bytesize', 'cancel_read', 'cancel_write',
'cd', 'close', 'closed', 'cts', 'dsr', 'dsrdtr', 'dtr', 'exclusive', 'fileno', 'flush', 'flushInput', 'flushOutput',
'getCD', 'getCTS', 'getDSR', 'getRI', 'getSettingsDict', 'get_settings', 'inWaiting', 'in_waiting',
'interCharTimeout', 'inter_byte_timeout', 'iread_until', 'isOpen', 'is_open', 'isatty', 'name', 'open',
'out_waiting', 'parity', 'port', 'portstr', 'read', 'read_all', 'read_until', 'readable', 'readall', 'readinto',
'readline', 'readlines', 'reset_input_buffer', 'reset_output_buffer', 'ri', 'rs485_mode', 'rts', 'rtscts', 'seek',
'seekable', 'sendBreak', 'send_break', 'setDTR', 'setPort', 'setRTS', 'set_buffer_size', 'set_output_flow_control',
'stopbits', 'tell', 'timeout', 'truncate', 'writable', 'write', 'writeTimeout', 'write_timeout', 'writelines',
'xonxoff']
```