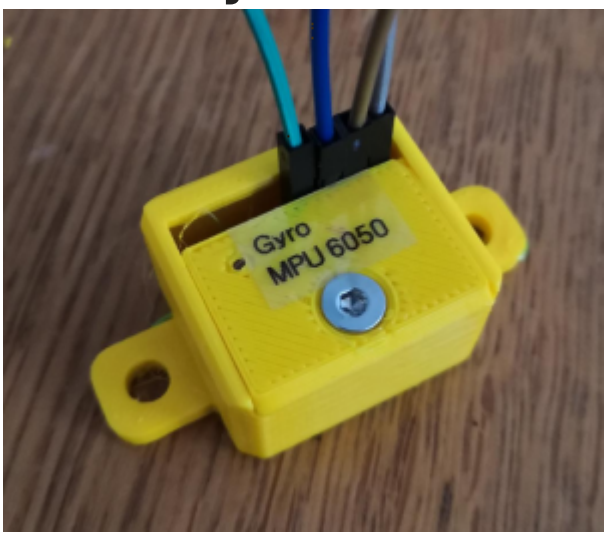
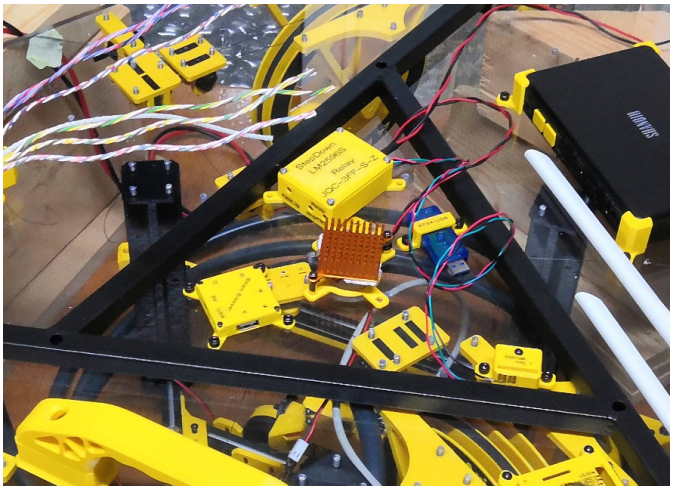
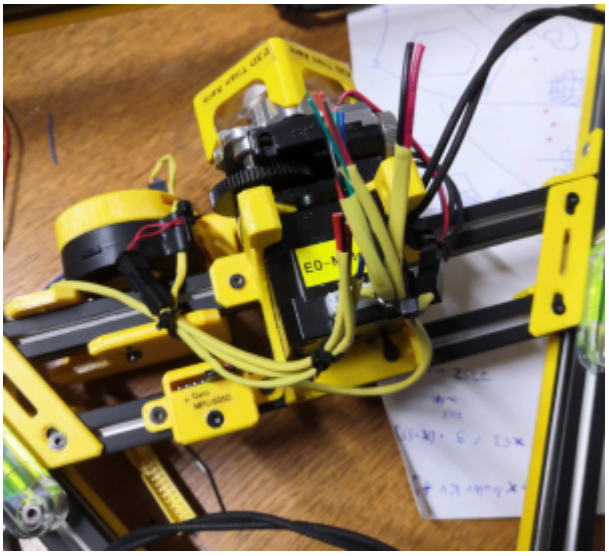
Monitoring and alerting | MPU 6050 (GY-521) Gyro + Accelerometer monitoring







Hardware → [Gyroscope and acceleration sensor GY-521 (MPU6050)] and basic configuration for Raspberry Pi

Features for Hangprinter

The MPU 6050 is a gyro which is used to measure room temperature and vibrations at the installation location. It helps to check for occured events like layer shifting by looking into the monitoring (InfluxDB + Grafana). It does not help to prevent them but is an experimental tool for analysing. Due to the very difficult exact positioning, a gyro is not suitable for longterm checking of absolute positions XYZ or absolute speeds, since the deviations add up over a short period of time. An angle error of even one degree will cause the estimated velocity to be off by some meters per seconds after a short period of elapsed time. After a single minute, one degree of angle error will cause the position estimate to be off by kilometers. For more details see

http://www.chrobotics.com/library/accel-position-velocity. For this reason an Inertial Measurement Unit MPU 9250 / GY-250 InvenSense was installed on effector side.

Having gyro sensors in Hangprinter means in theory that ...

1. the effector and frame can be better leveled when Hangprinter gets installed in a new location
2. the sensor helps with remote monitoring in the sense of simple motion tracking and emergency sensor for sensing out
   - tilts / misalignments of effector while printing caused by lost lines from bearings (shearing off), unevenly line tripping, obstacles, winding spools or blocking drives (possibly despite the closed loop system)
   - as soon as one this things or a mix of multiple causes happe, the sensor data will have really high and short peaks in relation to regular movements. Suche events usually are a mix of changed speed, acceleration and jerk, force and geometrical tilts
   - If the minimum or maximum tilts are exceeded this can trigger events like emergency halt or baby stepping ABCD drives to fix the effector tilt when printing
   - room vibrations (e.g. trams or trucks on the street) or other vibrations can be registered and visualized
3. analysing the surrounding temperature of the effector area is possible due to the simple fact that each GY-521 integrates a temperature sensor (could be as some kind of really cheap solution for fire hazard detection)

Wiring address

The MPU 6050 has primary address 0x68 or secondary address 0x69, depending on it's pin configuration

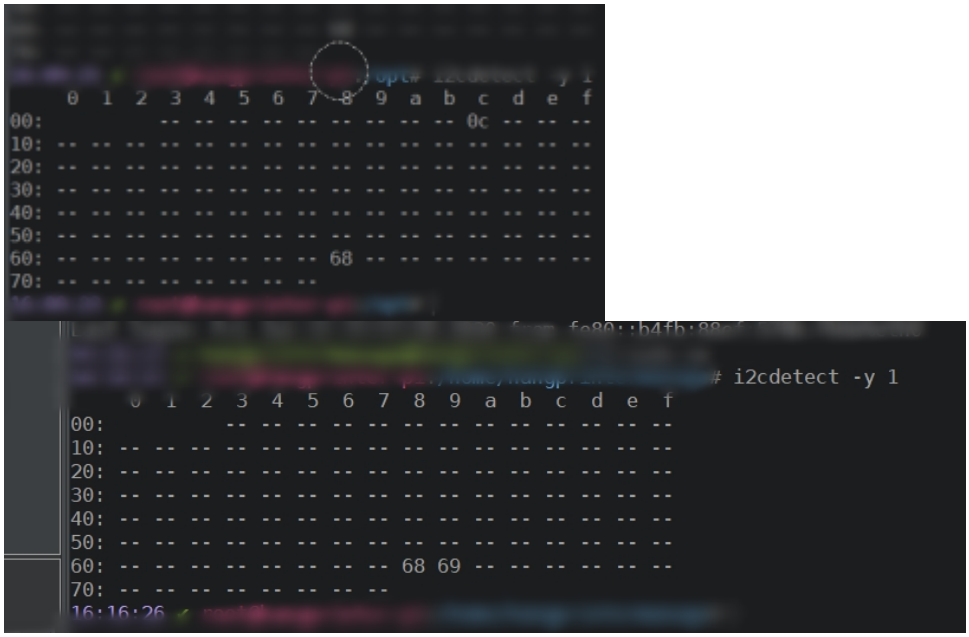
- 0x69

Troubleshooting

Module has wrong address

Module has wrong address

The MPU 6050 is a sensible I2C module which was connected to Raspberry Pi by slightly longer, unshielded cables. It figured out that there may occure bad side effects, for example if the main power chord (230 V) is to near to the cables. Then it can happen that the module gets a wrong address like 0x0c or that the module will fail by other unkown effects.



Python script to read the data

Trikarus uses a collectd script to collect the sensor data like regular system hardware metrics. This allows to visualize them in third party software. We use InfluxDB and Grafana to make graphs from sensor data and to make use of the data to perform printer relevant Repetier Server API calls.

Good sources to look at:

- https://www.twobitarcade.net/article/3-axis-gyro-micropython
- https://pypi.org/project/mpu6050-raspberrypi/#files
- https://github.com/KavinduZoysa/arduino/blob/master/MPU_6050/MPU_6050.ino

- https://how2electronics.com/measure-tilt-angle-using-mpu6050-gyro-accelerometer-arduino
- https://github.com/adamjezek98/MPU6050-ESP8266-MicroPython/blob/master/mpu6050.py
- https://github.com/rocheparadox/Kalman-Filter-Python-for-mpu6050/blob/master/AngleOMeter.py
- https://github.com/rocheparadox/Kalman-Filter-Python-for-mpu6050/blob/master/Kalman.py
- https://github.com/thisisG/MPU6050-I2C-Python-Class
- https://github.com/richardghirst/PiBits/tree/master/MPU6050-Pi-Demo

collectd Python plugin script

Preparations

Use Python3 interpreter! It's not tested with old legacy Python 2.X. Please not that this script does not work by calling it manually with "python mpu6050.py" because the register_* functions lead to execution error if not called by collectd.

```
#install collectd library
pip3.7 install collectd smbus

#fix a nasty uppercase/lowercase issue in that module
/usr/local/lib/python3.5/dist-packages/collectd.py
#change "from Queue import Queue, Empty"
#to "from queue import Queue, Empty"
```

Create plugin file

```
mkdir -p /opt/collectd_plugins
cd /opt/collectd_plugins
vim mpu6050.py
```

```
#!/usr/bin/python
import smbus
import math
```

```
import collectd

address=0x68
ADLOWHIGH=0
bus = smbus.SMBus(1) # bus = smbus.SMBus(0) for revision 1

def config_func(config):
    hw_address_set = False

    for node in config.children:
        key = node.key.lower()
        val = node.values[0]

        if key == 'adlowhigh':
            global ADLOWHIGH
            ADLOWHIGH = val
            hw_adress_set = True
        else:
            collectd.info('mpu6050 plugin: Unknown config key "%s"' % key)

    if hw_address_set:
        collectd.info('mpu6050 plugin: Using overridden address %s' % ADLOWHIGH)
    else:
        collectd.info('mpu6050 plugin: Using default address %s' % ADLOWHIGH)

def read_byte(reg):
    return bus.read_byte_data(address, reg)

def read_word(reg):
    h = bus.read_byte_data(address, reg)
    l = bus.read_byte_data(address, reg+1)
    value = (h << 8) + l
    return value

def read_word_2c(reg):
    val = read_word(reg)
    if (val >= 0x8000):
        return -((65535 - val) + 1)
    else:
        return val
```

```
def dist(a,b):
    return math.sqrt((a*a)+(b*b))

def get_y_rotation(x,y,z):
    radians = math.atan2(x, dist(y,z))
    return -math.degrees(radians)

def get_x_rotation(x,y,z):
    radians = math.atan2(y, dist(x,z))
    return math.degrees(radians)

def read_func():
    #collectd.info("gyro plugin: ADLOWHIGH = %s" % ADLOWHIGH)
    try:
    # https://medium.com/@kavindugimhanzoysa/lets-work-with-mpu6050-gy-521-part1-6db0d47a35e6
        scale_gyro = 131
        scale_accel = 16384.0
        earth_g = 9.80665 # m/s^2
        # collectd.info('gyro plugin: hardware address = %s' % address)
        bus = smbus.SMBus(1) # bus = smbus.SMBus(0) for revision 1
        if ADLOWHIGH == '0':
            address=0x68
        else:
            address=0x69
        bus.write_byte_data(0x68, 0x6b, 0) # activate the module to talk to it

        gyro_x = read_word_2c(0x43)
        gyro_y = read_word_2c(0x45)
        gyro_z = read_word_2c(0x47)

        # this converts the gyro values with deg/s unit
        gyro_x_scaled = gyro_x / scale_gyro
        gyro_y_scaled = gyro_y / scale_gyro
        gyro_z_scaled = gyro_z / scale_gyro

        accel_x = read_word_2c(0x3b)
        accel_y = read_word_2c(0x3d)
        accel_z = read_word_2c(0x3f)
```

```
        # this converts the gyro values with mm/s^2 unit
        accel_x_scaled = accel_x / scale_accel * earth_g
        accel_y_scaled = accel_y / scale_accel * earth_g
        accel_z_scaled = accel_z / scale_accel * earth_g

        temp = (read_word_2c(0x41) / 340) + 36.53

        # put values to collectd
        collectd.Values(plugin='mpu6050', plugin_instance='gyro0', type='gauge',
type_instance='temp', values=[temp]).dispatch()
        collectd.Values(plugin='mpu6050', plugin_instance='gyro0', type='gauge',
type_instance='gyro_x', values=[gyro_x]).dispatch()
        collectd.Values(plugin='mpu6050', plugin_instance='gyro0', type='gauge',
type_instance='gyro_y', values=[gyro_y]).dispatch()
        collectd.Values(plugin='mpu6050', plugin_instance='gyro0', type='gauge',
type_instance='gyro_z', values=[gyro_z]).dispatch()

        collectd.Values(plugin='mpu6050', plugin_instance='gyro0', type='gauge',
type_instance='gyro_x_scaled', values=[gyro_x_scaled]).dispatch()
        collectd.Values(plugin='mpu6050', plugin_instance='gyro0', type='gauge',
type_instance='gyro_y_scaled', values=[gyro_y_scaled]).dispatch()
        collectd.Values(plugin='mpu6050', plugin_instance='gyro0', type='gauge',
type_instance='gyro_z_scaled', values=[gyro_z_scaled]).dispatch()

        collectd.Values(plugin='mpu6050', plugin_instance='gyro0', type='gauge',
type_instance='accel_x', values=[accel_x]).dispatch()
        collectd.Values(plugin='mpu6050', plugin_instance='gyro0', type='gauge',
type_instance='accel_y', values=[accel_y]).dispatch()
        collectd.Values(plugin='mpu6050', plugin_instance='gyro0', type='gauge',
type_instance='accel_z', values=[accel_z]).dispatch()

        collectd.Values(plugin='mpu6050', plugin_instance='gyro0', type='gauge',
type_instance='accel_x_scaled', values=[accel_x_scaled]).dispatch()
        collectd.Values(plugin='mpu6050', plugin_instance='gyro0', type='gauge',
type_instance='accel_y_scaled', values=[accel_y_scaled]).dispatch()
        collectd.Values(plugin='mpu6050', plugin_instance='gyro0', type='gauge',
type_instance='accel_z_scaled', values=[accel_z_scaled]).dispatch()

        #collectd.info('gyro plugin: gyro_x = %s' % gyro_x)
        print("XYZ gyroscope:","{:.0f}".format(gyro_x)," (" , gyro_x_scaled,")
```

```
|","{:.0f}".format(gyro_y),"(",gyro_y_scaled,")
|","{:.0f}".format(gyro_z),"(",gyro_z_scaled,") | XYZ
acceleration:","{:.0f}".format(accel_x)," (" , accel_x_scaled,")
|","{:.0f}".format(accel_y),"(",accel_y_scaled,")
|","{:.0f}".format(accel_z),"(",accel_z_scaled,") | XY Rotations: " ,
get_x_rotation(accel_x_scaled, accel_y_scaled, accel_z_scaled), " | " ,
get_y_rotation(accel_x_scaled, accel_y_scaled, accel_z_scaled), " | temp ", temp)
    except Exception as e:
        collectd.error('mpu6050 plugin: exception: %s' % e)
        pass

collectd.register_config(config_func)
collectd.register_read(read_func,1) # read every 1 seconds
```

Configure collectd to add the script

```
vim /etc/collectd/collectd.conf
```

Add the following lines to the end of the config (or at the according place where Python plugins are put)

```
LoadPlugin python
<Plugin python>
    ModulePath "/opt/collectd_plugins"
    Import "mpu6050"
    <Module mpu6050>
    </Module>
</Plugin>
```

```
service collectd restart && journalctl -f -u collectd.service
```

Check out if values from collectd correctly flow into InfluxDB

```
# /opt/influxdb/influx
> USE collectd
> SHOW SERIES
> SHOW MEASUREMENTS
```

If you want to send a remote curl command, do the following

```
curl -cacert /etc/ssl/certs/site.de.ca.crt --cert /etc/ssl/certs/site.de.crt --key /etc/ssl/priv
```

> some example queries

```
SELECT last(value) FROM "mpu6050_value" WHERE "type_instance" = 'gyro_x' AND "host" =~
/^$host$/ AND $timeFilter GROUP BY time($interval)
SELECT last(value) FROM "mpu6050_value" WHERE "type_instance" = 'gyro_y' AND "host" =~
/^$host$/ AND $timeFilter GROUP BY time($interval)
SELECT last(value) FROM "mpu6050_value" WHERE "type_instance" = 'gyro_z' AND "host" =~
/^$host$/ AND $timeFilter GROUP BY time($interval)
SELECT last(value) FROM "mpu6050_value" WHERE "type_instance" = 'accel_x' AND "host" =~
/^$host$/ AND $timeFilter GROUP BY time($interval)
SELECT last(value) FROM "mpu6050_value" WHERE "type_instance" = 'accel_y' AND "host" =~
/^$host$/ AND $timeFilter GROUP BY time($interval)
SELECT last(value) FROM "mpu6050_value" WHERE "type_instance" = 'accel_z' AND "host" =~
/^$host$/ AND $timeFilter GROUP BY time($interval)
SELECT last(value) FROM "mpu6050_value" WHERE "type_instance" = 'temp' AND "host" =~ /^$host$/
AND $timeFilter GROUP BY time($interval)
```

Version #1
Erstellt: 2026-06-08 15:26:52 CEST von Mario Voigt
Zuletzt aktualisiert: 2026-06-08 15:30:15 CEST von Mario Voigt