3D Accelerometer (Beschleunigungs-Sensor Modul)

für den RP6 Robot, RP6 WIFI, ARUDINO und ASURO

Einleitung:

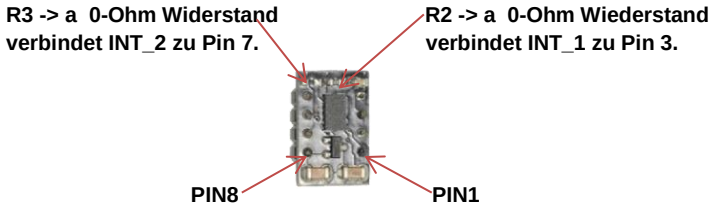
Allzweck 3D Beschleunigungssensor Modul für Roboter.
Mit dem Beschleunigungssensor Modul kann man leicht den absoluten Lagewinkel des Roboters mit einem Mikrokontroller bestimmen, z.B. Rollwinkel u. Neigewinkel. Außerdem kann man den Vektor eines Kollisionsimpules berechnen um dann entsprechend auszuweichen.

Hinweis: Oft werden die Begriffe Gyro (Drehratensensor) und Beschleunigungssensor vermischt. Hier eine Klarstellung:

Ein Gyro kann die Drehrate in Winkel pro Zeit messen, jedoch keinen absoluten Winkel. Dies kann nur der Beschleunigugssensor!

1.) PIN-Belegung des Accelerometer Module:

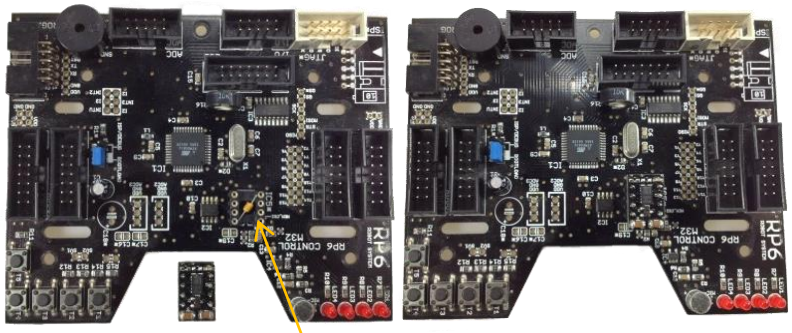
PIN1	CS/SPI enable	PIN5	SI/SDA
PIN2	SO/SDO (I2C address)	PIN6	SPC/SCL
PIN3	INT1	PIN7	INT2
PIN4	GND	PIN8	VCC



JM3 Engineering Inh. J. Maisel Dobelweg 18, D-85567 Grafing / Munich
Email: JHM@JM3-Engineering.de

2.) Anschluss an den RP6 Robot

Das Bild auf der linken Seite zeigt das M32 Kontroller-Board mit dem 8-Pin DIL Sockel.
Das Accelerometer-Modul passt ohne HW Anpassungen in den eingelöteten Sockel.



PIN3 und PIN 7 auf dem M32 Board sind nach Vdd verbunden! Die Widerstände R2 and R3 müssen deshalb entfernt werden!

Pin1 am DIL-Sockel befindet sich unten rechts. VDD ist 3.3 bzw. 5.0 V nominal.

3.) Anschluss an den RP6 WIFI, ASURO or ARUDINO

Alternativ kann das Modul auch an den I2C-Bus angeschlossen werden.

Dazu muss PIN1 des Modules an VDD angeschlossen werden. In diesem Fall ist PIN 5 der SDA PIN und PIN 6 ist SCL - mit PIN2 kann die Modul Adresse angepasst werden.
In diesem Fall kann dann das Modul zusammen mit RP6 WIFI, ASURO oder ARUDINO betrieben werden.

4.) Software Änderungen für das SPI Interface (SPI):

Der Datenzugriff und die Initialisierung wird über den SPI Bus abgewickelt – weitere Details der IC-Register finden Sie im Datenblatt des LIS302DLH.

Änderungen der Software des Kontroll-Boards:

1. RP6ControlLib.h:

```
/*************************************************************/
// SPI ACC Sensor
uint8_t ACCSENS_ReadByte(uint8_t regAddr);
void ACCSENS_ReadBytes(uint8_t startAddr, uint8_t* buffer, uint8_t length);
void ACCSENS_WriteByte(uint8_t regAddr, uint8_t data);
uint8_t ACCSENS_GetStatus();
```

2. RP6ControlLib.ccp

```
/*****************************************************************/
// SPI ACC Sensor  LIS302DLH

uint8_t cSPI::ACCSENS_ReadByte(uint8_t regAddr)
{
    regAddr |= 0x80;
    uint8_t data;

    PORTB &= ~ACC_CS;
    writeSPI(regAddr);
    data = readSPI();
    PORTB |= ACC_CS;

    return data;
}

// Reads "length" Bytes into the Buffer "buffer" from startAdr2 on

void cSPI::ACCSENS_ReadBytes(uint8_t startAddr, uint8_t* buffer, uint8_t length)
{
    startAddr |= 0xC0;

    PORTB &= ~ACC_CS;
    writeSPI(startAddr);
    readBufferSPI(buffer, length);
    PORTB |= ACC_CS;
}

// Write a single data byte to the specified sensor address

void cSPI::ACCSENS_WriteByte(uint8_t regAddr, uint8_t data)
{
    PORTB &= ~ACC_CS;
    writeSPI(regAddr);
    writeSPI(data);
    PORTB |= ACC_CS;
}

// Returns Sensor Status register - for checking if sensor is busy

uint8_t cSPI::ACCSENS_GetStatus()
{
    uint8_t status;

    PORTB &= ~ACC_CS;
    writeSPI(0x27);
    status = readSPI();
    PORTB |= ACC_CS;

    return status;
}
```

Das SW Beispiel zeigt wie auf das Modul zugegriffen werden kann.

3. Initialisierung des Sensors:

```
/*****************************************************************
 * initialize acceleration sensor module LIS302DLH
 * reg address MSB is auto-increment
 *
 * power up sensor, enable x, y, z axis
 * +/-2g scale, 50 Hz, no update while reading,
 * internal clock, all filter bypassed, Hpc = 1
 * CTRL_REG1:           1100 0111b:    Low Power Mode 10 Hz Output rate, update 50Hz,
                                       enable x, y, z axis
 * CTRL_REG2:           0000 0000b:    default
 * CTRL_REG3:           0000 0000b:    default
 * CTRL_REG4:           1100 1000b:    Block update, big endian, +/- 2g,
 * CTRL_REG5:           0000 0000b:    default
*****************************************************************/
```

Schreiben sie die folgenden Werte in die Register:

```
ACCSENS_WriteByte(0x20, 0xC7);
ACCSENS_WriteByte(0x21, 0x00);
ACCSENS_WriteByte(0x22, 0x00);
ACCSENS_WriteByte(0x23, 0x88);
ACCSENS_WriteByte(0x24, 0x00);
```

Register 0x21, 0x22, 0x24 und 0x25 müssen nicht geschrieben werden, falls die Standard Werte verwendet werden sollen.

4. Lesen der Sensor Daten:

1.) BYTE lesen:

```
buffer = ACCSENS_ReadByte(0x0F);
```

2.) Lesen der X- und y-achse(WORD):

```
buffer = ACCSENS_ReadByte(0x2A) | (ACCSENS_ReadByte(0x2B) << 8);
```

5.) Software Anpassungen zum Betrieb mit I2C Bus:

Der Datenzugriff und die Initialisierung wird über den I2C Bus abgewickelt – weitere Details der IC-Register finden Sie im Datenblatt des LIS302DLH.

5.1) Initialisierung des Sensors:

```
/*****************************************************************
 * initialize acceleration sensor module LIS302DLH
 * reg address MSB is auto-increment
 *
 * power up sensor, enable x, y, z axis
 * +/-2g scale, 50 Hz, no update while reading,
 * internal clock, all filter bypassed, Hpc = 1
 * CTRL_REG1:           1100 0111b:    Low Power Mode 10 Hz Output rate, update 50Hz,
                                       enable x, y, z axis
 * CTRL_REG2:           0000 0000b:    default
 * CTRL_REG3:           0000 0000b:    default
 * CTRL_REG4:           1100 1000b:    Block update, big endian, +/- 2g,
 * CTRL_REG5:           0000 0000b:    default
*****************************************************************/

I2CTWI_transmit4Bytes(ADR_ACC_MODULE, 0xA0, 0xC7, 0x00, 0x00);
// Slave Address; Write Register 0x20  - Data Reg1 – Data Reg 2, Data Reg3
I2CTWI_transmit3Bytes(ADR_ACC_MODULE, 0xA3, 0x88, 0x00);
// Slave Address; Write Register 0x23  - Data Reg4 – Data Reg 5
```

Die Slave-Adresse entspricht: 0x32 (PIN2 = High) oder 0x30 (PIN2 = Low)

5.2 Lesen der Sensor Daten vom I2C

```
/*****************************************************************/
// I2C ACC Sensor  LIS302DLH
// Read x axis acceleration data high and low Byte and put it into buffer

I2CTWI_readRegisters(ADR_ACC_MODULE, 0x29, &buffer[0], 1);
// Slave Address; Read Register 0x29; buffer; # of Bytes to read
I2CTWI_readRegisters(ADR_ACC_MODULE, 0x28, &buffer[1], 1);
// Slave Address; Read Register 0x28; buffer; # of Bytes to read
```

Der Sensor bietet eine ganze Reihe von Einstellungsmöglichkeiten – z.B. gibt es spezielle Filter die konfiguriert werden müssen – bitte lesen Sie die entsprechenden Stellen in der IC Spezifikation.