Der Beschleunigungssensor MPU6050

Zum Preis von ca. 5 bis 30 Euro kann man Module mit dem Beschleunigungssensor MPU6050 erstehen. Bei meiner Bestellung wurden zwei Steckerleisten (einmal gerade, einmal gewinkelt) mitgeliefert. Nach dem Anlöten einer solchen Steckerleiste lässt sich das Modul direkt an unsere Attiny-2.0-Platine anschließen. Das Modul besitzt einen Spannungsregulator, so dass es mit 5 V betrieben werden kann. Man beachte, dass nicht alle MPU6050-Module einen derartigen Spannungswandler besitzen. Auf dem Modul sind auch bereits die Pullup-Widerstände für die I2C-Kommunikation montiert, so dass die entsprechenden Jumper auf der Attiny-Platine nicht unbedingt gesetzt werden müssen.

Abbildung 1: Das Modul GY-521 mit dem Sensor MPU6050

Der Baustein digitalisiert zunächst die von den einzelnen Sensorkomponenten bezogenen Messwerte und legt sie in einem Register ab. Die Werte in diesem Register werden fortwährend synchron aktualisiert; die Aktualisierungsrate kann eingestellt werden. Die Werte können per I2C-Protokoll aus dem Register ausgelesen werden.

Die 7-Bit-Adresse des MPU6050 ist standardmäßig $68; bei einem Schreibvorgang ist die 8-Bit-Adresse dann 2*$68 = $D0, bei einem Lesevorgang $D1. Wenn man auf ein bestimmtes Register des Bausteins zugreifen möchte, muss man zuerst den Baustein adressieren und sodann die Nummer des Registers übermitteln.

Um z. B. einen Wert aus einem Register auszulesen, kann man die folgende BASCOM-Befehlsfolge benutzen:

```
I2cstart
I2cwbyte Mpu6050_adr              'Write-Adresse
If Err = 0 Then                   'ACK empfangen?
  I2cwbyte Regadr                 'Registerzeiger setzen
  I2cstart
  I2cwbyte Mpu6050_adr1           'Read-Adresse
  I2crbyte Ergebnis , Nack
End If
I2cstop
```

Will man mehrere aufeinander folgende Register auslesen, muss der Baustein nicht unbedingt neu adressiert werden und man kann auch auf das erneute Setzen des Registerzeigers verzichten. Dazu man beim Lesen nur ein ACK-Bit senden; dadurch wird der Registerzeiger nämlich automatisch um 1 erhöht:

```
  I2crbyte Ergebnis_1 , Ack
  I2crbyte Ergebnis_2 , Ack
  I2crbyte Ergebnis_3 , Ack
    ...
  I2crbyte Ergebnis_N , Nack
```

Der MPU650 besitzt ungefähr 100 Register. Einige von ihnen dienen wie oben schon erwähnt zur Übergabe von Messwerten. Die meisten haben aber Kontroll- und Steuerungsfunktion. So kann man beispielsweise die Aktualisierungsrate der Ergebnisregister oder die benutzte Skala einstellen. Detaillierte Informationen hierzu findet man in den MPU6050-Manuals. In der Standardeinstellung haben fast alle diese Register den Wert 0. Deswegen wollen wir uns im Rahmen dieser kleinen Einführung auch nur um zwei dieser Kontroll- und Steuerungsregister kümmern.

1. Das Register ***Power Management 1*** (Nr. $6B)

   Standardmäßig ist das SLEEP-Bit dieses Registers auf 1 gesetzt. In diesem Zustand kann der Baustein keine Messwerte liefern. Um den Baustein zu wecken, setzen wir das ganze Register auf $00.

2. Das Register ***Who Am I*** (Nr. $75)

   Diese Register ist vom Typ Read Only. Es beinhaltet den Wert $68, also die 7-Bit-Adresse des MPU6050-Bausteins. Das Register kann also zur Identifizierung und Funktionskontrolle eingesetzt werden.

Von den Ergebnisregistern wollen wir hier benutzen:

Adresse	Bezeichnung	Bedeutung
$3B	ACCEL_XOUT_H	Beschleunigung in x-Richtung (High-Byte)
$3C	ACCEL_XOUT_L	Beschleunigung in x-Richtung (Low-Byte)
$3D	ACCEL_YOUT_H	Beschleunigung in y-Richtung (High-Byte)
$3E	ACCEL_YOUT_L	Beschleunigung in y-Richtung (Low-Byte)
$40	ACCEL_ZOUT_H	Beschleunigung in x-Richtung (High-Byte)
$41	ACCEL_ZOUT_L	Beschleunigung in x-Richtung (High-Byte)
$42	TEMP_OUT_H	Temperatur (High-Byte)
$43	TEMP_OUT_L	Temperatur (Low-Byte)

An zwei Beispielen soll der Umgang mit dem MPU6050-Baustein erläutert werden.

Baustein identifizieren und Temperaturwerte auslesen

```
' Datei für Attiny-Platine von E. Eube, G. Heinrichs und U. Ihlefeldt
' Acceleration- und Gyro-Sensor mpu6050
' Ermittelt die Temperaturwerte als Dezimalzahl und gibt sie über Terminal (Text) aus
'--------------------------------------------------------------------------

$regfile = "attiny2313.dat"                                 'Attiny2313
$crystal = 4000000                                          '4 MHz
$baud = 9600


'*********************************************************
'****************** Deklarationen ************************

Declare Sub Mpu6050_write(regadr As Byte , Byval Wert As Byte)
Declare Function Mpu6050_read(regadr As Byte) As Byte

Dim Mpu6050_adr As Byte
Dim Mpu6050_adr1 As Byte
Dim Mpu6050_who_am_i As Byte
Dim Mpu6050_temp_h As Byte
Dim Mpu6050_temp_l As Byte
Dim Mpu6050_pwr_mgmt_1 As Byte

Dim Temp_h As Byte
Dim Temp_l As Byte
Dim Temp_int As Integer
Dim Temp_real As Single

Dim Antwort As Byte
Dim Ergebnis As Byte


'****************** Initialisierung ***********************

Ddrb = &B11111111                               'Port B als Ausgangsport
Ddrd = &B01110000                               'D4, D5, D6 als Ausgang; Rest als Eingang
Portd = &B10001111                              'Eingänge auf high legen
Waitms 50                                       'zum Laden des Kondensators bei Ta0

Config Scl = Portb.7
Config Sda = Portb.5

Mpu6050_adr = &HD0                               '&H68 ist 7-Bit-I2C-Adresse
Mpu6050_adr1 = &HD1
Mpu6050_who_am_i = &H75
Mpu6050_temp_h = &H41
Mpu6050_temp_l = &H42
Mpu6050_pwr_mgmt_1 = &H6B

'*********************************************************
'******************** Hauptprogramm **********************

Antwort = Mpu6050_read(mpu6050_who_am_i)         'Wer ist es?
If Antwort = &H68 Then Print "mpu6050 gefunden... "

Call Mpu6050_write(mpu6050_pwr_mgmt_1 , 0)      'aus dem sleep-modus holen

Do
  Wait 1
  Temp_h = Mpu6050_read(mpu6050_temp_h)
  Temp_l = Mpu6050_read(mpu6050_temp_l)
  Temp_int = 256 * Temp_h
  Temp_int = Temp_int + Temp_l
  Temp_real = Temp_int * 0.002941
  Temp_real = Temp_real + 36.53                 'vgl. S. 31 von RM_MPU-6000A.pdf

  Print Temp_real
Loop
```

```
'*********************************************************
'****************** Unterprogramme ***********************

Sub Mpu6050_write(regadr As Byte , Byval Wert As Byte)
  I2cstart
  I2cwbyte Mpu6050_adr
  If Err = 0 Then
    I2cwbyte Regadr
    I2cwbyte Wert
  End If
  I2cstop
End Sub

Function Mpu6050_read(regadr As Byte) As Byte
  Ergebnis = 0
  I2cstart
  I2cwbyte Mpu6050_adr
  If Err = 0 Then                                       'ACK empfangen
    I2cwbyte Regadr
    I2cstart
    I2cwbyte Mpu6050_adr1
    I2crbyte Ergebnis , Nack
  End If
  I2cstop
  Mpu6050_read = Ergebnis
End Function
```

Es fällt auf, dass die Temperaturwerte nach dem Einschalten leicht ansteigen; offensichtlich heizt sich der Baustein im Betrieb leicht auf.

Beschleunigungswerte in X-Richtung anzeigen

```
' Datei für Attiny-Platine von E. Eube, G. Heinrichs und U. Ihlefeldt
' Acceleration- und Gyro-Sensor mpu6050
' Ermittelt die Beschleunigungswerte (Accel_out_x) in m/s2
' und gibt sie über Terminal (Text) aus
'-------------------------------------------------------------------------------

$regfile = "attiny2313.dat"                             'Attiny2313
$crystal = 4000000                                      '4 MHz
$baud = 9600


'*********************************************************
'****************** Deklarationen ************************

Declare Sub Mpu6050_write(regadr As Byte , Byval Wert As Byte)
Declare Function Mpu6050_read(regadr As Byte) As Byte

Dim Mpu6050_adr As Byte
Dim Mpu6050_adr1 As Byte
Dim Mpu6050_who_am_i As Byte
Dim Mpu6050_accel_xout_h As Byte
Dim Mpu6050_accel_xout_l As Byte
Dim Mpu6050_pwr_mgmt_1 As Byte

Dim Accel_x_h As Byte
Dim Accel_x_l As Byte
Dim Accel_x_int As Integer
Dim Accel_x_real As Single

Dim Antwort As Byte
Dim Ergebnis As Byte


'****************** Initialisierung **********************
```

```
Ddrb = &B11111111                                   'Port B als Ausgangsport
Ddrd = &B01110000                                   'D4, D5, D6 als Ausgang; Rest als Eingang
Portd = &B10001111                                  'Eingänge auf high legen
Waitms 50                                           'zum Laden des Kondensators bei Ta0

Config Scl = Portb.7
Config Sda = Portb.5

Mpu6050_adr = &HD0                                  '&H68 ist 7-Bit-I2C-Adresse
Mpu6050_adr1 = &HD1
Mpu6050_who_am_i = &H75
Mpu6050_accel_xout_h = &H3B
Mpu6050_accel_xout_l = &H3C
Mpu6050_pwr_mgmt_1 = &H6B

'*********************************************************
'******************** Hauptprogramm **********************

Antwort = Mpu6050_read(mpu6050_who_am_i)
If Antwort = &H68 Then Print "mpu6050 gefunden... "

Call Mpu6050_write(mpu6050_pwr_mgmt_1 , 0)       'aus dem sleep-modus holen

Do
  Wait 1
  Accel_x_h = Mpu6050_read(mpu6050_accel_xout_h)
  Accel_x_l = Mpu6050_read(mpu6050_accel_xout_l)
  Accel_x_int = 256 * Accel_x_h
  Accel_x_int = Accel_x_int + Accel_x_l

  '1 g entspricht 2^14 = 16384 bei Standard-Range 2g; also Beschl = accel_x_int*9.81/16384

  Accel_x_real = Accel_x_int * 0.00059875

  Print Accel_x_real ; "m/s2 "
Loop

'*********************************************************
'******************** Unterprogramme *********************

Sub Mpu6050_write(regadr As Byte , Byval Wert As Byte)
  I2cstart
  I2cwbyte Mpu6050_adr
  If Err = 0 Then
    I2cwbyte Regadr
    I2cwbyte Wert
  End If
  I2cstop
End Sub

Function Mpu6050_read(regadr As Byte) As Byte
  Ergebnis = 0
  I2cstart
  I2cwbyte Mpu6050_adr
  If Err = 0 Then                                    'ACK empfangen
    I2cwbyte Regadr
    I2cstart
    I2cwbyte Mpu6050_adr1
    I2crbyte Ergebnis , Nack
  End If
  I2cstop
  Mpu6050_read = Ergebnis
End Function
```

Genauigkeit der gemessenen Beschleunigungswerte

Vor der Messung wurde der Baustein zunächst so positioniert, dass seine x-Richtung parallel zum Gravitationsfeld lag (g_{max}). Während der Messung wurde der Baustein so gedreht, dass

seine x-Richtung erst senkrecht (g_0) und dann antiparallel zum Gravitationsfeld lag (g_{min}).

In der Senkrecht-Lage sollte der Wert 0 angezeigt werden; in Wirklichkeit finden wir für g_0 einen Wert von ca. 0,5 m/s^2. In der Tat entnimmt man dem Manual, dass bei 0 g (Zero-g) eine Abweichung von 50/1000 von 1 g auftauchen kann, das sind 5% von 9,81 m/s^2, also ca. 0,5 m/s^2.

Für g_{max} finden wir einen Wert von ca. 10,3 m/s^2 und für g_{min} einen von -9,5 m/s^2. Ziehen wir zur Korrektur der Nullpunktsabweichung diesen Wert g_0 von g_{max} bzw. g_{min} ab, so erhalten wir fast identische Beträge: 9, 8 m/s^2 bzw. 10,0 m/s^2. Die Abweichungen liegen unterhalb der vom Hersteller angegebenen Kalibrationstoleranz von 3%.

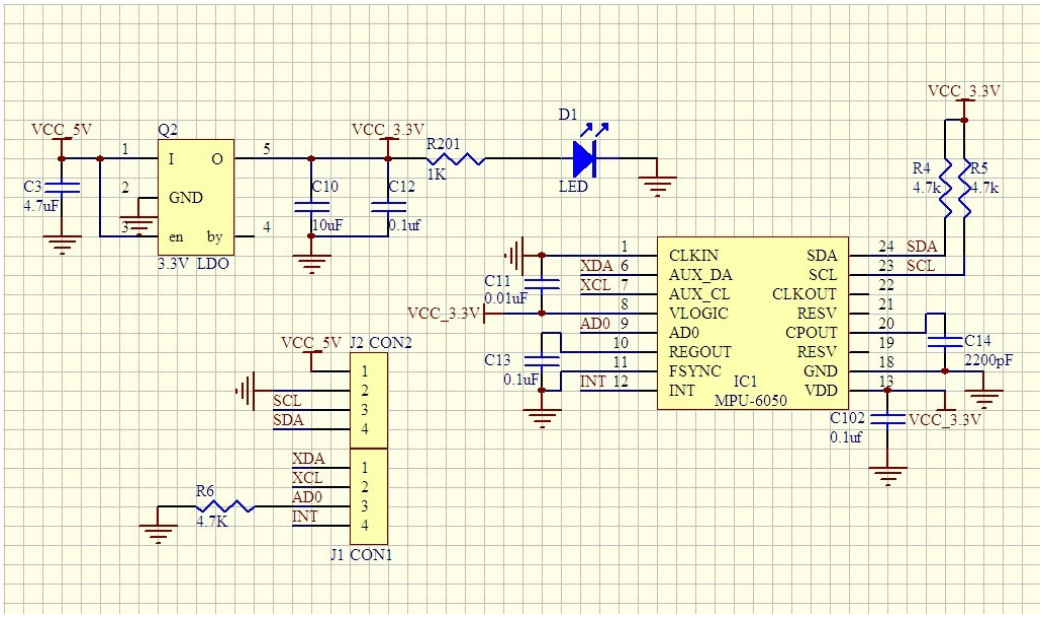

Abbildung 2: Schaltbild des GY-521-Moduls
(Quelle: http://arduino.cc/playground/uploads/Main/MPU6050-V1-SCH.jpg)