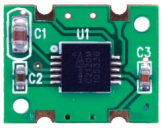


Vielleicht weil das Beschleunigungssensor- Modul von ELV so ein Verkaufsschlager war, hat Pollin seit kurzem ein günstiges Magnetsensor-Modul namens HDMM01 im Angebot, das sich als elektronischer Kompass verwenden lässt. Auf dem Modul befindet sich ein MMC2120 von MEMSIC. Der Sensor misst in zwei Achsen mit einer Auflösung von 12 Bit zwischen ±2 Gauss und verfügt über eine I²C Schnittstelle, so dass er sich leicht an einen Microcontroller anbinden lässt. Natürlich hab ich mir gleich eins von den Modulen geordert und ein wenig damit rumgespielt.

Anschluss | Der Anschluss des Moduls an einen AVR ist - I²C sei Dank - kaum der Rede wert: zwei Pins des Moduls dienen der Stromversorgung (der Sensor mag lt. Datenblatt alles zwischen 2.7 und 5.25 V), die anderen beiden Pins sind SDA und SCL und werden dementsprechend direkt mit dem µC verbunden, I²C Pull-ups nicht vergessen.

I²C Kommunikation | Der Sensor hat fest eingestellt die Adresse &h60. Das LSB der Adresse gibt bei I²C bekanntlich an, ob geschrieben oder gelesen werden soll, dementsprechen ist die Schreibadresse &h60, die Leseadresse ist &h61. Das Initiieren einer Messung und das Auslesen des Ergebnisses geschieht folgendermaßen: man erzeugt eine Startbedingung (SDA geht auf low während SCL high ist) und adressiert den Sensor zum Schreiben (&h60). Der Sensor hat nur ein Register, das an der Adresse &h00 liegt, man schreibt also diese Adresse auf den Bus, um im nächsten Schritt das Register manipulieren zu können. Das Setzen des LSB in diesem Register löst eine Messung aus, man überträgt dementsprechend &h01 und erzeugt eine Stopp-Bedingung (SDA geht auf high während SCL high ist). Nun genehmigt man dem Sensor einige Millisekunden Pause um die eigentliche Messung durchzuführen. Um die Daten abzuholen, wird zunächst wieder eine Start-Bedingung auf dem Bus erzeugt, dann teilt man dem Sensor mit, welches Datenbyte man lesen will. Dazu wird die Schreibadresse übertragen (&h60) und dann der Index des Datenbytes. Hier sollen alle fünf Bytes (Kontrollregister und Daten), also von Index 0 gelesen werden, man schreibt somit &h00 auf den Bus. Nach einer erneuten Startbedingung legt man die Leseadresse (&h61) auf den Bus und kann sich dann in Folge fünf Bytes abholen, deren ersten vier mit einen Acknowledge quittiert werden. Nach dem fünften Byte erzeugt man eine Stopp-Bedingung.

Datenauswertung | Der Sensor löst die Messung für die x- und y-Dimension jeweils mit zwölf Bit auf. Dementsprechend ist ein Messwert auf zwei Bytes verteilt. Das erste gelesene Byte ist das Kontrollregister, also beginnen die Messdaten ab dem zweiten Byte. Das zweite gelesene

Byte ist high-Byte des x-Wertes, das dritte gelesene Byte dessen low-Byte. Das vierte Byte ist high-Byte des y-Wertes und das fünfte Byte low-Byte des y-Wertes. Das ungenutzte obere Nibble der high-Bytes soll man laut Datenblatt maskieren (soweit ich sehe, ist es aber sowieso immer 0). Man braucht jetzt nur noch jeweils high- und low-Byte verrechnen, um an den entsprechenden Datenpunkt zu kommen.

Code | Folgend ein kleines Stückchen BASCOM-Code, der kontinuierlich das Messen initiiert, die Messwerte ausliest und per UART versendet - allerdings nicht in physikalsichen Einheiten, sondern so, wie der Sensor die Daten codiert.

{codecitation class="brush: plain; gutter: true;toolbar: false" width="509px"}' beispielprogramm zum auslesen des MMC2120MG
' malte ahlers 2011
' weitere infos auf malteahlers.de
'
'
'
' compiler steuerung
$regfile = "m8def.dat"
$crystal = 8000000
$framesize = 64
$swstack = 64
$hwstack = 64
$baud = 9600
$lib "i2c_twi.lbx"
'
' hardware konfigurieren
' - taster
Config Pinb.6 = Input
Portb.6 = 1
Config Pinb.7 = Input
Portb.7 = 1
Config Pind.5 = Input
Portd.5 = 1
Taster1 Alias Pinb.6
Taster2 Alias Pinb.7
Taster3 Alias Pind.5
' - leds
Config Portd.7 = Output
Config Portd.6 = Output
Config Portb.0 = Output
Led_gl Alias Portd.7

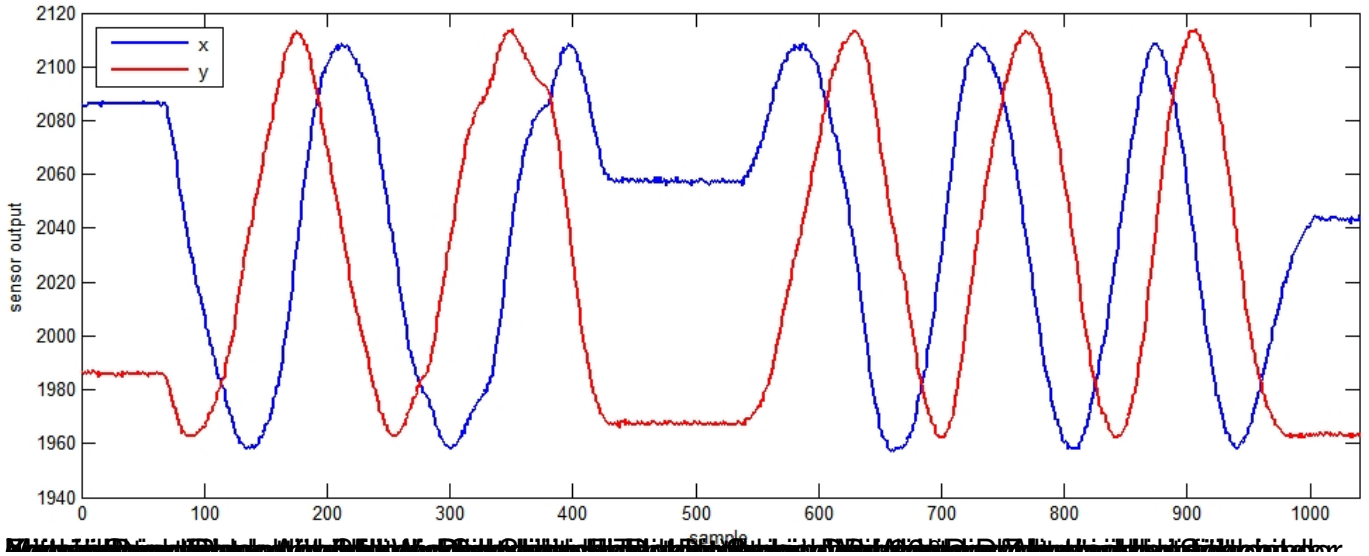
```
Led_rt Alias Portd.6
Led_gr Alias Portb.0
' - i2c
Config Twi = 400000
Config Scl = Portc.5
Config Sda = Portc.4
'
' konstanten
Const Cmp_w = &H60
Const Cmp_r = &H61
'
' variablen
Dim I As Byte
Dim Dat(5) As Byte
Dim X As Word
Dim Y As Word
'
' hauptprog
I2cinit
'
Do
 '
 ' * set/ reset coil *
 I2cstart
 '
 ' sensor adressieren (schreiben)
 I2cwbyte Cmp_w
 '
 ' register adressieren
 I2cwbyte &H00
 '
 ' register manipulieren -> set coil
 I2cwbyte &B00000010
 '
 I2cstop
 '
 Waitms 1
 '
 I2cstart
 '
 ' sensor adressieren (schreiben)
 I2cwbyte Cmp_w
 ' register adressieren
 I2cwbyte &H00
 '
 ' register manipulieren -> reset coil
```

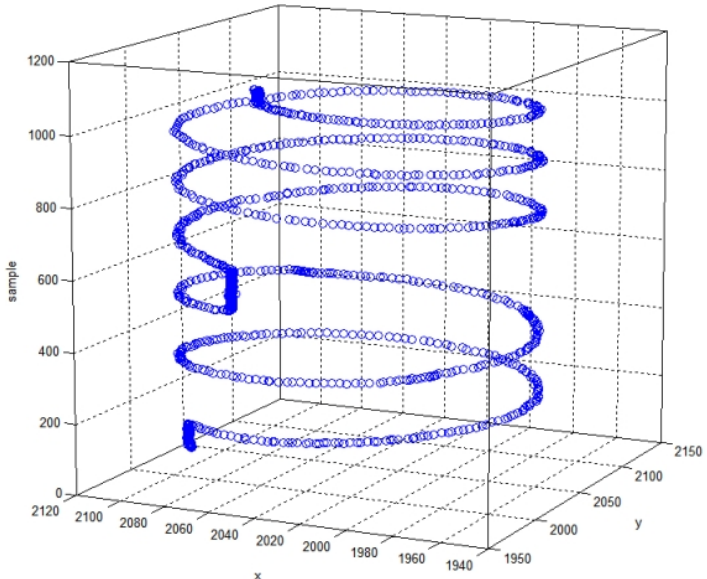
```
I2cwbyte &B00000100
'
I2cstop
'
Waitms 5
'
' * messung *
I2cstart
'
' sensor adressieren (schreiben)
I2cwbyte Cmp_w
'
' register adressieren
I2cwbyte &H00
'
' register manipulieren -> messung
I2cwbyte &H01
'
I2cstop
'
Waitms 5
'
I2cstart

I2cwbyte Cmp_w
'
' leseindex setzen
I2cwbyte &H00
'
I2cstart
'
' sensor adressieren (lesen)
I2cwbyte Cmp_r
'
For I = 1 To 4
' bytes 1-4 holen mit ack
I2crbyte Dat(i) , Ack
Next
'
' byte 5 holen mit nack
I2crbyte Dat(5) , Nack
'
I2cstop
'
' oberes nibble im high bytes löschen
Dat(2) = Dat(2) And &B0000_1111
```

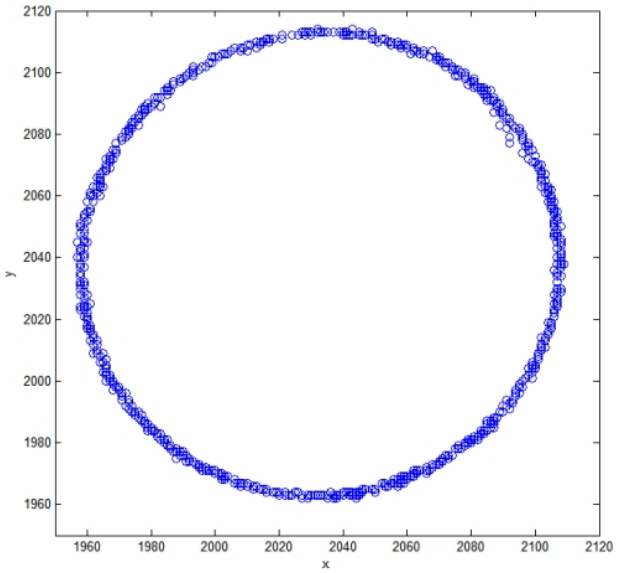
```
 Dat(4) = Dat(4) And &B0000_1111
 '
 ' x-wert aus high- und lowbyte berechnen
 X = Dat(2) * 256
 X = X + Dat(3)
 '
 ' y-wert aus high- und lowbyte berechnen
 Y = Dat(4) * 256
 Y = Y + Dat(5)
 '
 Print X ; " " ; Y
 '
 Waitms 40
 '
 Toggle Led_rt
 '
Loop
End{/codecitation}
```

Hier das BASCOM-File zum Download: mmc2120mg.bas

Messung | Hier ein Beispiel für die Rohdaten einer solchen Messnung auf Basis des obigen Codebeispiels. Ich habe meinen Testaufbau einfach auf meinen Schreibttischdrehstuhl gelegt und zwei Runden in eine Richtung gedreht, nach kurzer Pause dann drei Runden in Gegenrichtung. Dabei wurde bei ca. 20 Hz kontinuierlich gemessen und die Messdaten per serieller Schnittstelle vom AVR auf meinen PC überragen. Trägt man nun den x- und y-Wert gegen den Sampleindex (also im Prinzip die Zeit) auf, erhält man folgendes Diagramm:


x
y
sensor output
sample


sample
x
y


x
y