

# Software-Entwicklung für eingebettete Systeme

Prof. Dr. Peter Gerwinski

11. Mai 2017

# Inhalt

1. Was sind eingebettete Systeme?
2. Arrays und Pointer für Fortgeschrittene
3. Einführung in Unix
4. TCP/IP-Netzwerke in der Praxis
5. Bus-Systeme

... und *viel* praktisches Arbeiten!

Exkursion zu Lufthansa (Frankfurt) – wann?

# 5 Bus-Systeme

## 5.1 Was sind Bus-Systeme?

*Ein Bus ist ein System zur Datenübertragung zwischen mehreren Teilnehmern über einen gemeinsamen Übertragungsweg.*

[https://de.wikipedia.org/wiki/Bus\\_\(Datenverarbeitung\)](https://de.wikipedia.org/wiki/Bus_(Datenverarbeitung))

Beispiele:

- Computer kommuniziert mit Peripherie
- Computer kommunizieren (direkt) miteinander
- Prozessor kommuniziert mit externem Speicher
- Teile eines Prozessors kommunizieren miteinander

# 5 Bus-Systeme

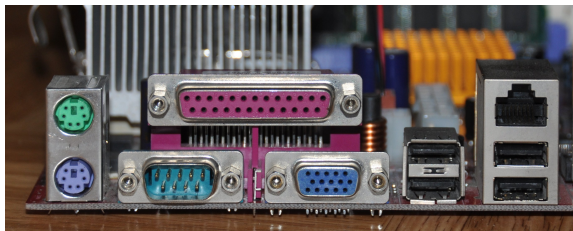
## 5.1 Was sind Bus-Systeme?

Standard-Computer:

- Einsteckkarten: PCI (und Vorgänger)
- Festplatten: SATA (und Vorgänger)
- USB, FireWire, ...
- Ethernet, CAN-Bus, ...
- WLAN, BlueTooth, IR, ...
- PS/2, RS-232, Centronics

Minimal-Hardware:

- RS-232
- I<sup>2</sup>C (TWI)
- SPI



# 5 Bus-Systeme

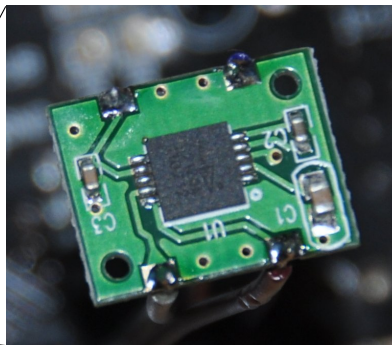
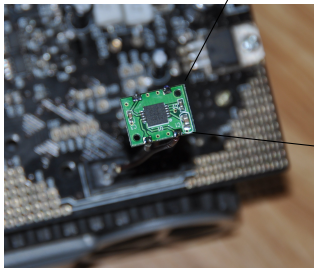
## 5.1 Was sind Bus-Systeme?

Standard-Computer:

- Einsteckkarten: PCI (und Vorgänger)
- Festplatten: SATA (und Vorgänger)
- USB, FireWire, ...
- Ethernet, CAN-Bus, ...
- WLAN, BlueTooth, IR, ...
- PS/2, RS-232, Centronics

Minimal-Hardware:

- RS-232
- I<sup>2</sup>C (TWI)
- SPI



# 5 Bus-Systeme

## 5.1 Was sind Bus-Systeme?

<i>seriell</i>	jedes Bit einzeln übertragen
<i>parallel</i>	mehrere Bits gleichzeitig
<i>synchron</i>	Abgleich über Steuerleitung: <i>Takt</i>
<i>asynchron</i>	Abgleich über Zeitvereinbarungen
<i>Punkt-zu-Punkt</i>	genau zwei Teilnehmer
<i>busfähig</i>	mehrere Teilnehmer, mit <i>Adressierung</i>

- I<sup>2</sup>C: seriell, synchron, mit Adressierung
- RS-232: seriell, asynchron, Punkt-zu-Punkt
- RS-485, USB, CAN: seriell, asynchron, mit Adressierung
- SPI: seriell, synchron, Punkt-zu-Punkt oder mit Adressierung

# 5 Bus-Systeme

## 5.2 RS-232

seriell

- *TX*: 1 Leitung für Daten
- *RX*: ggf. 1 Leitung für Daten in der anderen Richtung
- *GND*: gemeinsame *Masse*
- evtl. zusätzliche Steuerleitungen

synchron

- *keine* Taktleitung für Abgleich, wann Daten anliegen
- Stattdessen: Abgleich über Zeitvereinbarungen

→ Jeder Teilnehmer braucht eine eigene Zeitbasis.

Punkt-zu-Punkt

- nur 2 Teilnehmer vorgesehen

## 5.2 RS-232

Synchronisation

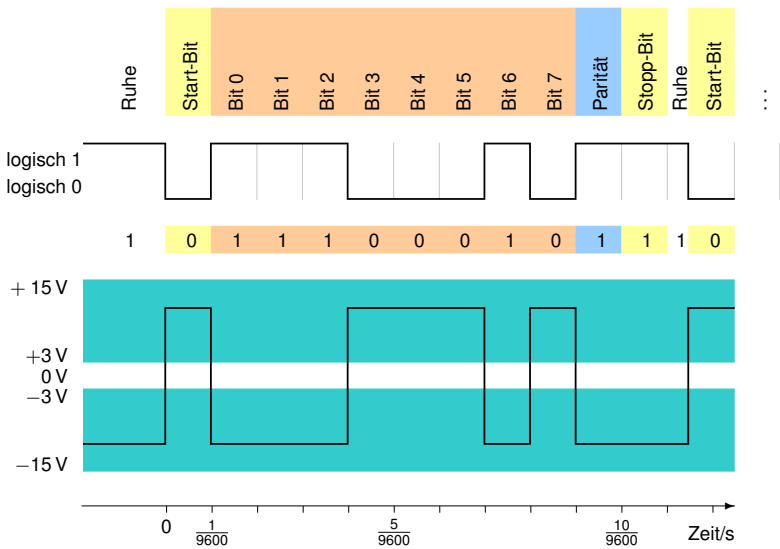
Daten

Check

9600 Baud, 8 Daten-Bits, ungerade Parität, 1 Stopp-Bit

Beispiel-Daten: ASCII „G“ = 71 = 0100 0111 binär

Übertragung der Daten von rechts (Bit 0) nach links (Bit 7)





# 5 Bus-Systeme

## 5.3 I<sup>2</sup>C (TWI)

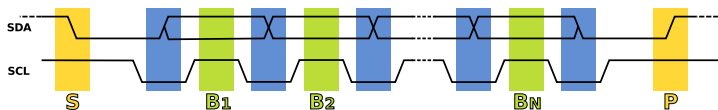
I<sup>2</sup>C = Inter-IC; TWI = Two-Wire-Interface

seriell

- *SDA*: 1 Leitung für Daten (in beiden Richtungen)
- *SCL*: Taktleitung (Clock)
- *GND*: gemeinsame Masse
- evtl. *VCC*: Stromversorgung für Peripheriegerät

synchron

- Abgleich über Taktleitung



busfähig

- *Master* initiiert Kommunikation und steuert Taktleitung
- erstes gesendetes Byte: *Adresse* des Teilnehmers
- 2 Adressen pro Teilnehmer: Lesen/Schreiben

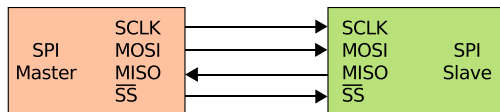
# 5 Bus-Systeme

## 5.4 SPI

### Serial Peripheral Interface

seriell

- *MISO*: Master Out, Slave In
- *MOSI*: Master In, Slave Out
- *SCLK*: Taktleitung (Clock)
- $\overline{SS}$ : Slave Select (invertiert)
- *GND*: gemeinsame Masse
- evtl. *VCC*: Stromversorgung für Peripheriegerät



synchron

- Abgleich über Taktleitung

busfähig

- *Master* initiiert Kommunikation und steuert Taktleitung
- *Slave* wird über *Slave Select* ausgewählt

# 5 Bus-Systeme

## 5.4 SPI

### Serial Peripheral Interface

seriell

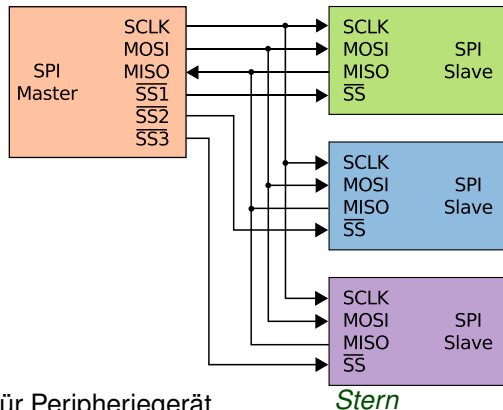
- *MISO*: Master Out, Slave In
- *MOSI*: Master In, Slave Out
- *SCLK*: Taktleitung (Clock)
- $\overline{SS}$ : Slave Select (invertiert)
- *GND*: gemeinsame Masse
- evtl. *VCC*: Stromversorgung für Peripheriegerät

synchron

- Abgleich über Taktleitung

busfähig

- *Master* initiiert Kommunikation und steuert Taktleitung
- *Slave* wird über *Slave Select* ausgewählt



# 5 Bus-Systeme

## 5.4 SPI

Serial Peripheral Interface

seriell

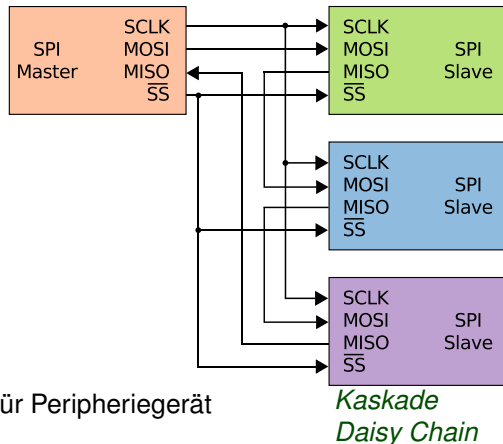
- *MISO*: Master Out, Slave In
- *MOSI*: Master In, Slave Out
- *SCLK*: Taktleitung (Clock)
- $\overline{SS}$ : Slave Select (invertiert)
- *GND*: gemeinsame Masse
- evtl. *VCC*: Stromversorgung für Peripheriegerät

synchron

- Abgleich über Taktleitung

busfähig

- *Master* initiiert Kommunikation und steuert Taktleitung
- *Slave* wird über *Slave Select* ausgewählt



# 5 Bus-Systeme

## 5.4 SPI

Serial Peripheral Interface

seriell

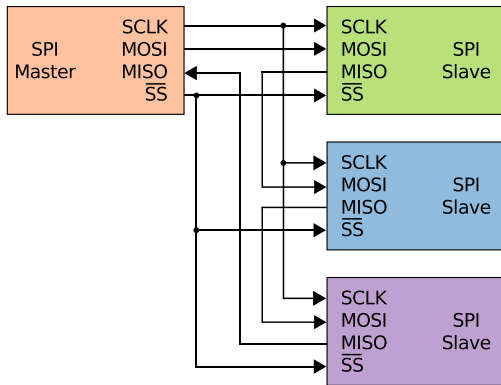
- *MISO*: Master Out, Slave In
- *MOSI*: Master In, Slave Out
- *SCLK*: Taktleitung (Clock)
- $\overline{SS}$ : Slave Select (invertiert)
- *GND*: gemeinsame Masse
- evtl. *VCC*: Stromversorgung für Peripheriegerät

synchron

- Abgleich über Taktleitung

busfähig

- *Master* initiiert Kommunikation und steuert Taktleitung
- *Slave* wird über *Slave Select* ausgewählt



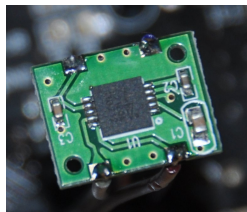
*Kaskade  
Daisy Chain*

Slave gibt MOSI-Input um 1 Takt verzögert an MISO aus → Master setzt „im richtigen Moment“  $\overline{SS}$

# 5 Bus-Systeme

## 5.5 Beispiel: Benutzung des I<sup>2</sup>C-Busses

```
void read_compass (uint16_t *x, uint16_t *y)
{
    I2CTWI_transmit2Bytes (0x60, 0x00, 0x02); // set coil
    mSleep (1);
    I2CTWI_transmit2Bytes (0x60, 0x00, 0x04); // reset coil
    mSleep (5);
    uint8_t result[5];
    I2CTWI_transmit2Bytes (0x60, 0x00, 0x01); // Messung starten
    mSleep (5); // 5ms warten, bis Sensor fertig gemessen hat
    I2CTWI_transmitByte (0x60, 0x01); // Leseindex setzen
    I2CTWI_readBytes (0x61, result, 4); // lesen: msb x, lsb x, msb y, lsb y
    result[0] &= 0b00001111; // Unwichtige Bits vom msb abschneiden
    result[2] &= 0b00001111;
    *x = (result[0] << 8) + result[1]; // Wert berechnen aus msb und lsb
    *y = (result[2] << 8) + result[3];
}
```

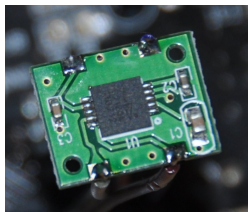


# 5 Bus-Systeme

## 5.5 Beispiel: Benutzung des I<sup>2</sup>C-Busses

```
void read_compass (uint16_t *x, uint16_t *y)
{ ... }
```

```
int main (void)
{
    uint16_t compass_x, compass_y;
    read_compass (&compass_x, &compass_y);
    writeInteger (compass_x, DEC);
    writeChar ('_');
    writeInteger (compass_y, DEC);
    writeChar ('\n');
    return 0;
}
```

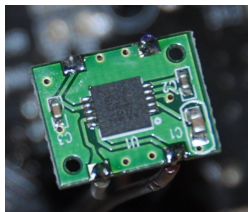


# 5 Bus-Systeme

## 5.5 Beispiel: Benutzung des I<sup>2</sup>C-Busses

```
void read_compass (uint16_t *x, uint16_t *y)
{ ... }
```

```
int main (void)
{
    uint16_t compass_x, compass_y;
    read_compass (&compass_x, &compass_y);
    writeInteger (compass_x, DEC);
    writeChar ('_');
    writeInteger (compass_y, DEC);
    writeChar ('\n');
    return 0;
}
```

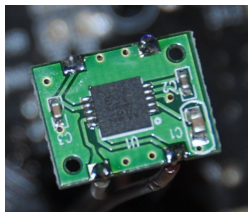




# 5 Bus-Systeme

## 5.5 Beispiel: Benutzung des I<sup>2</sup>C-Busses

```
void read_compass (uint16_t *x, uint16_t *y)
{
    I2CTWI_transmit2Bytes (0x60, 0x00, 0x02); // set coil
    mSleep (1);
    I2CTWI_transmit2Bytes (0x60, 0x00, 0x04); // reset coil
    mSleep (5);
    uint8_t result[5];
    I2CTWI_transmit2Bytes (0x60, 0x00, 0x01); // Messung starten
    mSleep (5); // 5ms warten, bis Sensor fertig gemessen hat
    I2CTWI_transmitByte (0x60, 0x01); // Leseindex setzen
    I2CTWI_readBytes (0x61, result, 4); // lesen: msb x, lsb x, msb y, lsb y
    result[0] &= 0b00001111; // Unwichtige Bits vom msb abschneiden
    result[2] &= 0b00001111;
    *x = (result[0] << 8) + result[1]; // Wert berechnen aus msb und lsb
    *y = (result[2] << 8) + result[3];
}
```

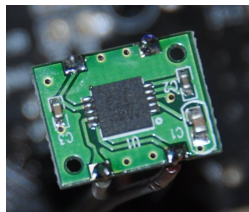


# 5 Bus-Systeme

## 5.5 Beispiel: Benutzung des I<sup>2</sup>C-Busses

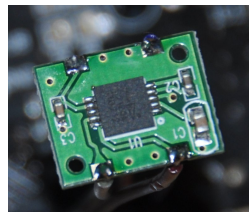
```
void read_compass (uint16_t *x, uint16_t *y)
{
    I2CTWI_transmit2Bytes (0x60, 0x00, 0x02); // set coil
    mSleep (1);
    I2CTWI_transmit2Bytes (0x60, 0x00, 0x04); // reset coil
    mSleep (5);
    uint8_t result[5];
    I2CTWI_transmit2Bytes (0x60, 0x00, 0x01); // Messung starten
    mSleep (5); // 5ms warten, bis Sensor fertig gemessen hat
    I2CTWI_transmitByte (0x60, 0x01); // Leseindex setzen
    I2CTWI_readBytes (0x61, result, 4); // lesen: msb x, lsb x, msb y, lsb y
    result[0] &= 0b00001111; // Unwichtige Bits vom msb abschneiden
    result[2] &= 0b00001111;
    *x = (result[0] << 8) + result[1]; // Wert berechnen aus msb und lsb
    *y = (result[2] << 8) + result[3];
}
```

Gerät adressieren:  
Schreib-Adresse



# 5 Bus-Systeme

## 5.5 Beispiel: Benutzung des I<sup>2</sup>C-Busses

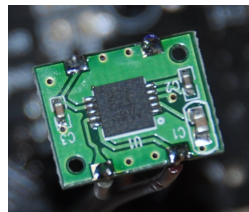


```
void read_compass (uint16_t *x, uint16_t *y)
{
    I2CTWI_transmit2Bytes (0x60, 0x00, 0x02); // set coil
    mSleep (1);
    I2CTWI_transmit2Bytes (0x60, 0x00, 0x04); // reset coil
    mSleep (5);
    uint8_t result[5];
    I2CTWI_transmit2Bytes (0x60, 0x00, 0x01); // Messung starten
    mSleep (5); // 5ms warten, bis Sensor fertig gemessen hat
    I2CTWI_transmitByte (0x60, 0x01); // Leseindex setzen
    I2CTWI_readBytes (0x61, result, 4); // lesen: msb x, lsb x, msb y, lsb y
    result[0] &= 0b00001111; // Unwichtige Bits vom msb abschneiden
    result[2] &= 0b00001111;
    *x = (result[0] << 8) + result[1]; // Wert berechnen aus msb und lsb
    *y = (result[2] << 8) + result[3];
}
```

Gerät adressieren:  
Schreib-Adresse  
Register-Nr.  
innerhalb des Geräts

# 5 Bus-Systeme

## 5.5 Beispiel: Benutzung des I<sup>2</sup>C-Busses



```
void read_compass (uint16_t *x, uint16_t *y)
{
    I2CTWI_transmit2Bytes (0x60, 0x00, 0x02); // set coil
    mSleep (1);
    I2CTWI_transmit2Bytes (0x60, 0x00, 0x04); // reset coil
    mSleep (5);
    uint8_t result[5];
    I2CTWI_transmit2Bytes (0x60, 0x00, 0x04); // Messung starten
    mSleep (5); // 5ms warten, bis Sensor fertig gemessen hat
    I2CTWI_transmitByte (0x60, 0x01); // Leseindex setzen
    I2CTWI_readBytes (0x61, result, 4); // lesen: msb x, lsb x, msb y, lsb y
    result[0] &= 0b00001111; // Unwichtige Bits vom msb abschneiden
    result[2] &= 0b00001111;
    *x = (result[0] << 8) + result[1]; // Wert berechnen aus msb und lsb
    *y = (result[2] << 8) + result[3];
}
```

Gerät adressieren:  
Schreib-Adresse

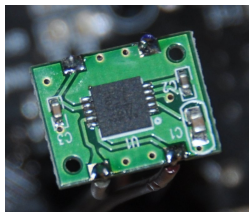
Register-Nr.  
innerhalb des Geräts

Befehl: Spule an

# 5 Bus-Systeme

## 5.5 Beispiel: Benutzung des I<sup>2</sup>C-Busses

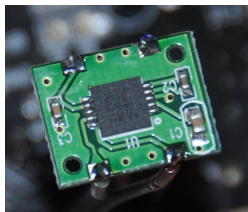
```
void read_compass (uint16_t *x, uint16_t *y)
{
    I2CTWI_transmit2Bytes (0x60, 0x00, 0x02); // set coil
    mSleep (1);
    I2CTWI_transmit2Bytes (0x60, 0x00, 0x04); // reset coil
    mSleep (5);
    uint8_t result[5];
    I2CTWI_transmit2Bytes (0x60, 0x00, 0x01); // Messung starten
    mSleep (5); // 5ms warten, bis Sensor fertig gemessen hat
    I2CTWI_transmitByte (0x60, 0x01); // Leseindex setzen
    I2CTWI_readBytes (0x61, result, 4); // lesen: msb x, lsb x, msb y, lsb y
    result[0] &= 0b00001111; // Unwichtige Bits vom msb abschneiden
    result[2] &= 0b00001111;
    *x = (result[0] << 8) + result[1]; // Wert berechnen aus msb und lsb
    *y = (result[2] << 8) + result[3];
}
```



# 5 Bus-Systeme

## 5.5 Beispiel: Benutzung des I<sup>2</sup>C-Busses

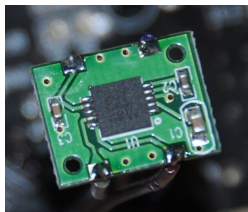
```
void read_compass (uint16_t *x, uint16_t *y)
{
    I2CTWI_transmit2Bytes (0x60, 0x00, 0x02); // set coil
    mSleep (1);
    I2CTWI_transmit2Bytes (0x60, 0x00, 0x04); // reset coil
    mSleep (5);
    uint8_t result[5];
    I2CTWI_transmit2Bytes (0x60, 0x00, 0x01); // Messung starten
    mSleep (5); // 5ms warten, bis Sensor fertig gemessen hat
    I2CTWI_transmitByte (0x60, 0x01); // Leseindex setzen
    I2CTWI_readBytes (0x61, result, 4); // lesen: msb x, lsb x, msb y, lsb y
    result[0] &= 0b00001111; // Unwichtige Bits vom msb abschneiden
    result[2] &= 0b00001111;
    *x = (result[0] << 8) + result[1]; // Wert berechnen aus msb und lsb
    *y = (result[2] << 8) + result[3];
}
```



# 5 Bus-Systeme

## 5.5 Beispiel: Benutzung des I<sup>2</sup>C-Busses

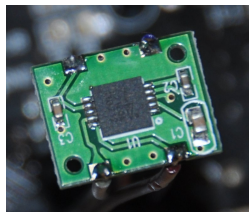
```
void read_compass (uint16_t *x, uint16_t *y)
{
    I2CTWI_transmit2Bytes (0x60, 0x00, 0x02); // set coil
    mSleep (1);
    I2CTWI_transmit2Bytes (0x60, 0x00, 0x04); // reset coil
    mSleep (5);
    uint8_t result[5];
    I2CTWI_transmit2Bytes (0x60, 0x00, 0x01); // Messung starten
    mSleep (5); // 5ms warten, bis Sensor fertig gemessen hat
    I2CTWI_transmitByte (0x60, 0x01); // Leseindex setzen
    I2CTWI_readBytes (0x61, result, 4); // lesen: msb x, lsb x, msb y, lsb y
    result[0] &= 0b00001111; // Unwichtige Bits vom msb abschneiden
    result[2] &= 0b00001111;
    *x = (result[0] << 8) + result[1]; // Wert berechnen aus msb und lsb
    *y = (result[2] << 8) + result[3];
}
```



# 5 Bus-Systeme

## 5.5 Beispiel: Benutzung des I<sup>2</sup>C-Busses

```
void read_compass (uint16_t *x, uint16_t *y)
{
    I2CTWI_transmit2Bytes (0x60, 0x00, 0x02); // set coil
    mSleep (1);
    I2CTWI_transmit2Bytes (0x60, 0x00, 0x04); // reset coil
    mSleep (5);
    uint8_t result[5];
    I2CTWI_transmit2Bytes (0x60, 0x00, 0x01); // Messung starten
    mSleep (5); // 5ms warten, bis Sensor fertig gemessen hat
    I2CTWI_transmitByte (0x60, 0x01); // Leseindex setzen
    I2CTWI_readBytes (0x61, result, 4); // lesen: msb x, lsb x, msb y, lsb y
    result[0] &= 0b00001111; // Unwichtige Bits vom msb abschneiden
    result[2] &= 0b00001111;
    *x = (result[0] << 8) + result[1]; // Wert berechnen aus msb und lsb
    *y = (result[2] << 8) + result[3];
}
```





# 5 Bus-Systeme

## 5.5 Beispiel: Benutzung des I<sup>2</sup>C-Busses

Messung:

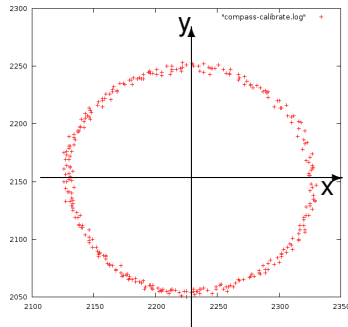
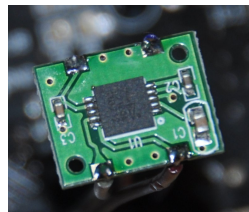
Roboter im Kreis drehen lassen,  
x- und y-Werte aufzeichnen  
und gegeneinander auftragen

Ergebnis: Ellipse

Anwendung:

Roboter in eine Himmelsrichtung ausrichten

- Achsen normieren:  
Mittelwert subtrahieren
- Nulldurchgang einer Achse  
markiert eine Himmelsrichtung
- Vorzeichen der anderen Achse  
sagt aus, welche Himmelsrichtung



Beispiel: nach Norden ausrichten

Grob drehen, bis y positiv ist, fein drehen, bis x = 0 ist