

Programming of Sensors on “Arduino”-Board with digital and analogue Output

Tutor: *Mr. Göran Dustler*

Place: *Norra Station
Norra Stationsgatan 6A
28148 Hässleholm*

Date: *03-02-2009 - 03-27-2009*

Program: *Student Exchange*

University: *University of Rostock
18055 Rostock, Germany*

Students: *Stephan Schedler
Sven Poeggel
Maik Gotzmann
Andy Seidel
Ruben Koch
Karsten Wiedmann*

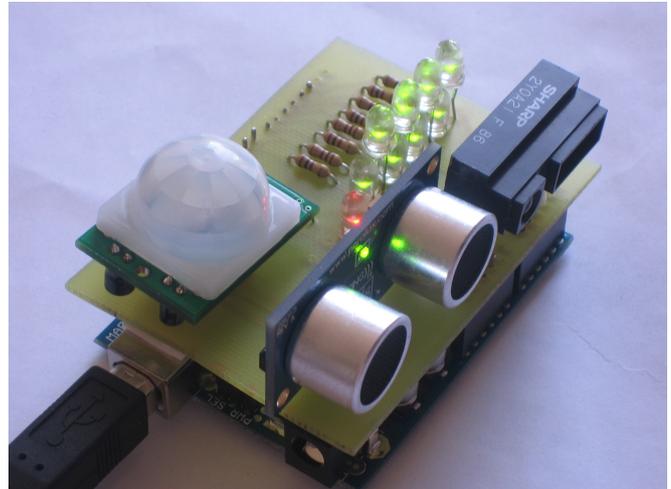
I. Directory

I. Directory.....	I
1 Motion-Distance-Board.....	1
1.1 Introduction / Description.....	1
1.2 PIN-Allocation / Circuit Diagram.....	2
1.3 Sensors.....	3
1.3.1 PIR Sensor.....	3
1.3.2 PING Ultrasonic Range Finder.....	5
1.3.3 Sharp GP2Y0A21YK0F.....	7
2 Acceleration-Compass-Board.....	9
2.1 Introduction / Description.....	9
2.2 PIN-Allocation / Circuit Diagram.....	10
2.3 Sensors.....	11
2.3.1 MMA7260QT 3-Axis.....	11
2.3.2 Memsic 2125 Dual-Axis.....	13
2.3.3 HM55B Compass Module.....	16
3 Display-Board.....	20
3.1 Introduction / Description.....	20
3.2 PIN-Allocation / Circuit Diagram.....	20
3.3 Programming.....	21
3.4 Miscellaneous.....	22
4 Helpful Information.....	23
4.1 Arduino Pin Mapping.....	23
4.2 Output via Hyperterminal.....	23
II. Bibliography.....	II

1 Motion-Distance-Board

1.1 Introduction / Description

The “Motion-Distance-Board” contains two different kinds of sensors, which allows the student an easy entrance in programming sensors on “Arduino”-Board. It includes a simply digital sensor for detecting motion in its environment. The distance could be measured in two different ways: via an infrared (Sharp) and an ultrasonic (PING) distance sensor. The two sensors should

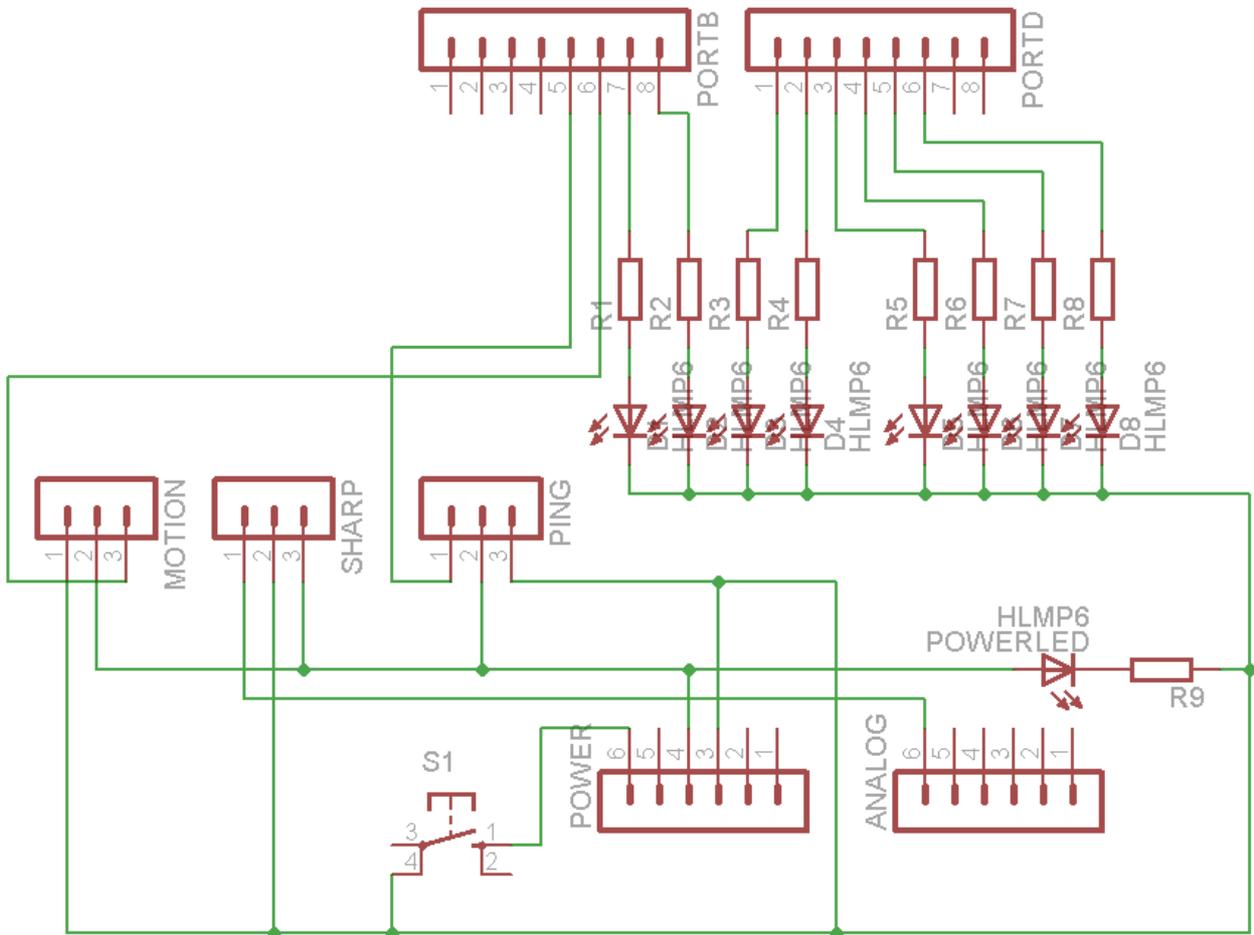


explain the difference between analogue and digital values. The Sharp and the PING sensor are next to each other to compare their calculated distances.

For the visual output there are 8 LEDs arranged in a row to give out the distance or the detection of a motion using the PIR sensor. In addition there is a Power-LED and a reset switch on the board.

The following documentation gives an overview how to program the sensors. The category “programming” explains an example procedure and could be used as help. There are also different ways of programming. It is recommended to implement different techniques to get an better understanding for this issue.

1.2 PIN-Allocation / Circuit Diagram



Analogue I/O	Digital I/O		
PC0	PD2-7 & PB0-1	PB2	PB3
SHARP	LED	PIR	PING

1.3 Sensors

1.3.1 PIR Sensor

Introduction

The PIR Sensor is a device that detects motion of its environment. It allows solid handling and simple programming and is ideal for an entrance in programming sensors. It has one digital output and shows motion by sending a high signal.

The following description shows only an overview of the PIR Sensor.

For better understanding the reading of the datasheet is recommendend (["http://www.parallax.com/dl/docs/prod/audiovis/PIRSensor-V1.1.pdf"](http://www.parallax.com/dl/docs/prod/audiovis/PIRSensor-V1.1.pdf)).

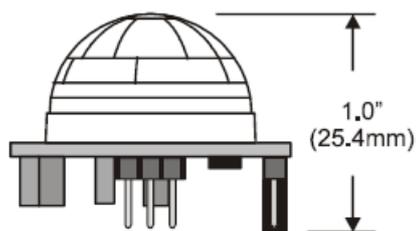


Abb 1. PIR Sensor [Paralax 08: PIR]

Functionality

The PIR measures infrared levels emitted by surrounding objects. It is a pyroelectric device, which includes a crystalline element. This material generates an electric charge by exposing infrared radiation. An amplifier measures this changing in connection to the infrared radiation. If the infrared signal changes rapidly, the amplifier will react with a high signal on the output pin. The high signal lasts for a couple of seconds, then it will be return to low signal.

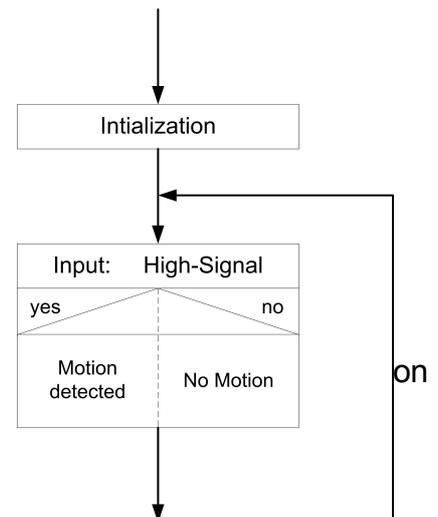
For working correctly the PIR Sensor needs a certain time for initialization between 10-60seconds for learning its environment. In this time there should be as little motion as possible to calibrate the sensor. In this time the output shows a low signal.

The PIR Sensor has a range of approximately 20 feet (ca. 6m), which vary with the condition of the environment.

Programming

The programming of the PIR can be realized in a very simple (but inefficient) way, by polling. Polling means a continuous reading of the input by an endless loop. For reading the I/O pin PB2 of the “Arduino”-board should be declared as input. The loop consists of only one IF-statement, which checks the Input of a high signal. If a high signal is detected the motion can be signaled by switching a LED or by using the Hyperterminal.

An example is programmed in “Motion_digital”.



Miscellaneous / helpful Information

The PIR sensor has a very high sensitivity, which sometimes makes the testing difficult. So it is recommended to cover the sensor to ensure that no motion (low-level) could be detected. After signaling motion and sending a high level for a couple of seconds the sensor sends a second short high level. This is a property of the sensor and could be disabled by the programming, for example by using a delay.

A better efficient way to program is the use of interrupts. At this point should only be referred to this technique.

1.3.2 PING Ultrasonic Range Finder

Introduction

The Parallax PING sensor is an ultrasonic based distance sensor for non-contact distance measurements. It is very easy to connect to arduino because it requires only one I/O pin which has to switch from output to input during the measurement. The data sheet will be found on the following link: <http://www.zerko.ch/downloads/28015pingv1.5.pdf>

Functionality

The Ping sensor works by transmitting an ultrasonic "chirp" signal well above the human hearing range. The chirp signal is also used by dolphins and bats and it has a high resistance to interference.

By measuring the echo pulse width, the distance to the target can be calculated.

The sensor works in a range from 2cm to 3m and the burst indicator LED shows the activity.

The pin definition is very easy because of using one digital I/O pin only. You have to connect GND, the power (+5 VDC) and signal. The I/O pin for the PING on the board is PB3.

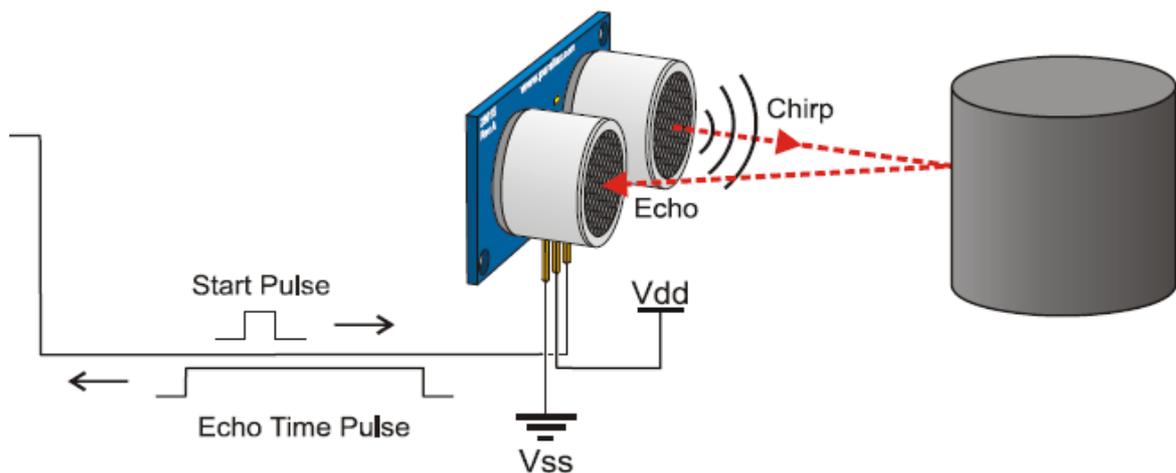


Abb 2. PING [Paralax 08: PING]

Programming

To measure the distance with the PING you send out an impulse (which is typical 5 μ s but minimal 2 μ s long), switch the PIN from OUT to Input and wait for the echo. With the clock frequency and pulse width of the received signal, which is the time the echo needs to come back to the transceiver/receiver, you can calculate the distance to the object you aimed, but pay attention that the speed of sound in the air depends on the temperature.

The output of the distance can be realized by using the LEDs on the board or the hyperterminal which is included in Microsoft Windows.

Miscellaneous

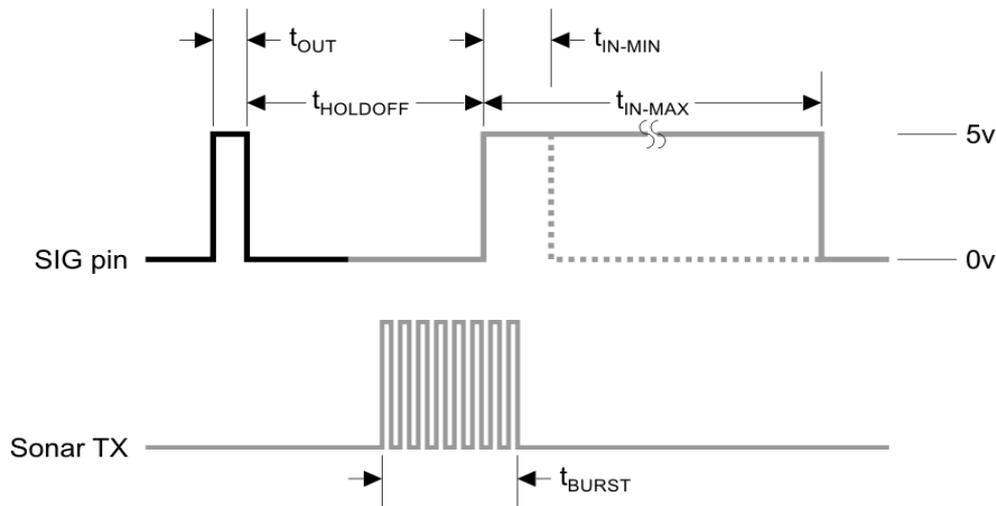


Abb 3. Impulsduration [Paralax 08: PING]

- you can calculate the speed of sound in the air as a function of temperature by the following formula: $C = 331,5 + (0,6 * T_c)$ m/s
- you have to pay attention that the measured time between sending and receiving the ultrasonic signal pass through the same distance two times
- before you can send out the burst you have to make sure that the Output is LOW so first set LOW on the Output before setting it HIGH
- the burst has to be 2 μ s (minimum) long but 5 μ s is typical
- activate the Pull-up resistor of the I/O pin you are using
- if you are using the 8 bit timer you limit the maximum distance range of the sensor

1.3.3 Sharp GP2Y0A21YK0F

Introduction

The Sharp GP2Y0A21YK0F is a non-contact distance measuring sensor, based on infrared.

Functionality

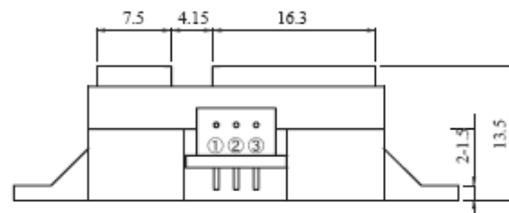
The sensor contains a PSD (position sensitive detector) , an IRED (infrared emitting diode) and a signal processing circuit.

The IRED emits throughout infrared light and the light will be reflected by any item. The PSD waits for infrared light and will receive it when the item is in a range from 10 to 80 cm.

The signal processing unit converts the signal and then the device outputs the (analogue) voltage corresponding to the detection distance.

To initiate the sensor it is required to put one ground connection and one 5V connection on it. The third connection delivers the analogue Output.

The variety of the reflectivity of the object, the environmental temperature and the operating duration are not influenced easily to the distance detection because of adopting the triangulation method.



Connector signal

signal name
① Vo
② GND
③ Vec

Connector :
J.S.T.TRADING COMPANY,LTD,
S3B-PH

Abb 4. Sharp GP2Y0A021YK and package dimension [Sharp-World 08: GP2Y0A021YK]

Programming

The sensor provides an analogue signal which you have to convert to a digital signal. Therefore you have to use the AD-converter which is integrated in the ATmega168. After digitalization you should convert the binary number to the corresponding value to get a range.

Now you can write it out using the Hyperterminal or using the LEDs.

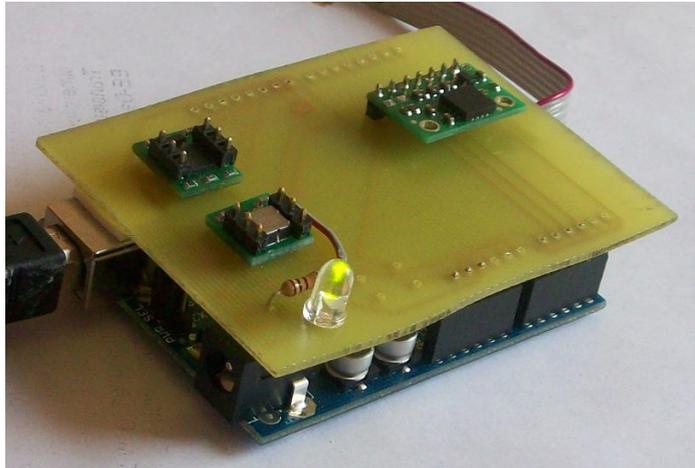
Miscellaneous / helpful Information

- The non-linear characteristic curve of the sensor is difficult to handle (exponential). It is recommended to ease it by 2 or more linear curves. The more curves you take, the more raises the accuracy.
- It is recommended to limit the range of the sensor from 10 to 80 cm to eliminate implausible values.

2 Acceleration-Compass-Board

2.1 Introduction / Description

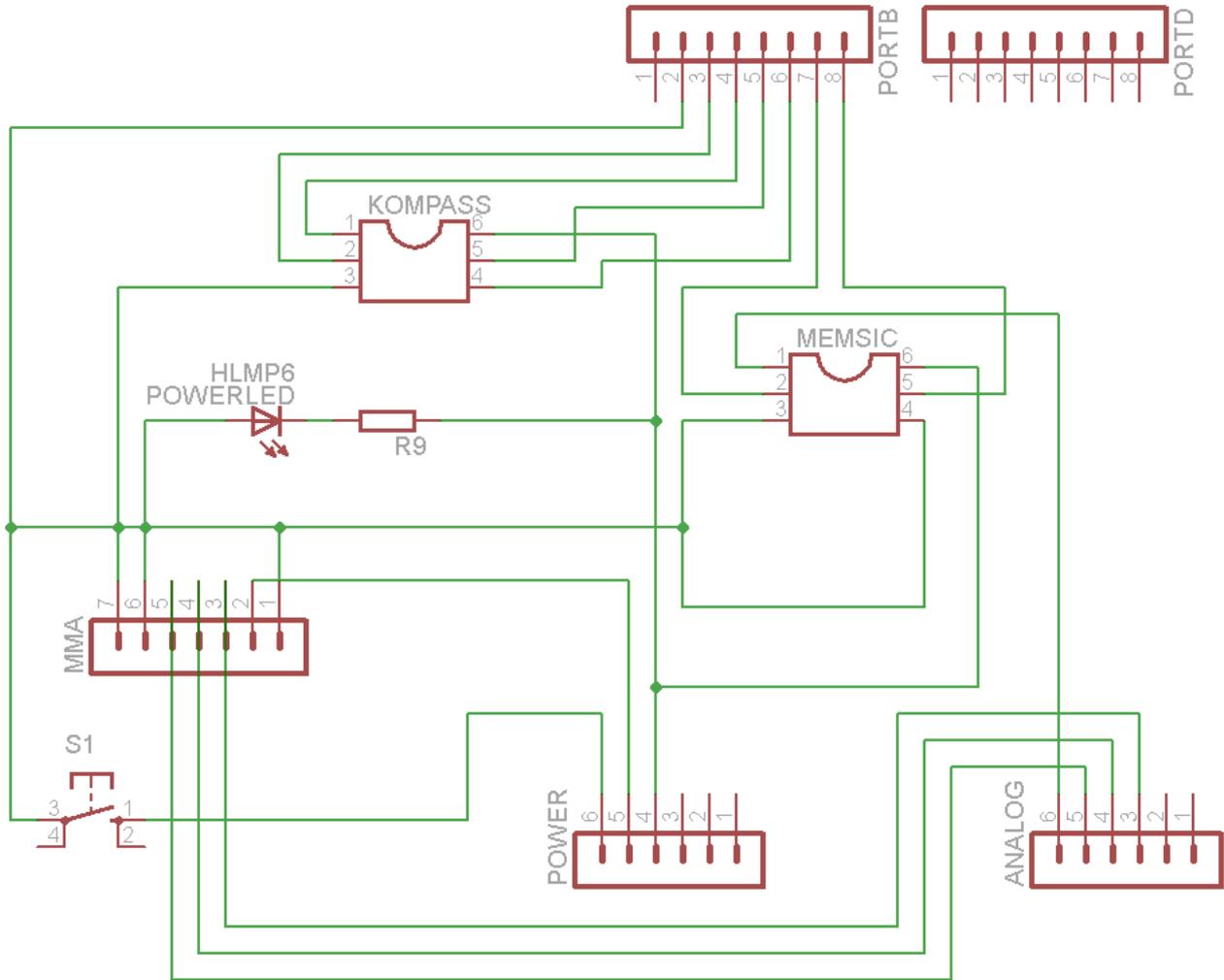
The “Acceleration-Compass-Board” board is equipped with two acceleration sensors (the Memsic and the MMA), a compass module, a temperature sensor which is included in the Memsic, a Power-LED and a reset switch. The Board offers a wide-ranging use of programming Sensors, starting by simply analogue temperature measurement till compass points measurement by communicating via a protocol.



The Output could be realized by using the Hyperterminal. It is also possible to connect the “Display”-Board for displaying the Output on its LED-Matrix. This allows a supplemental training of programming a microcontroller.

The following documentation gives an overview how to program the sensors. The category “programming” explains an example procedure and could be used as help. There are also different ways of programming. It is recommended to implement different techniques to get an better understanding for this issue.

2.2 PIN-Allocation / Circuit Diagram



Analogue I/O			
PC0	PC1	PC2	PC3
MEMSIC Temp.	MMA Z-Axis	MMA Y-Axis	MMA X-Axis

Digital I/O					
PB0	PB1	PB2	PB3	PB4	PB5
MEMSIC X-Axis	MEMSIC Y-Axis	COMPASS CLK	COMPASS /EN	COMPASS Dout	COMPASS Din

2.3 Sensors

2.3.1 MMA7260QT 3-Axis

Introduction

The MMA7260QT is a triple-axis accelerometer. It is a great IC, but its small, leadless package makes it difficult for the typical student or hobbyist to use. The device also operates at 2.2 V to 3.6 V, which can make interfacing difficult for microcontrollers operating at 5 V.

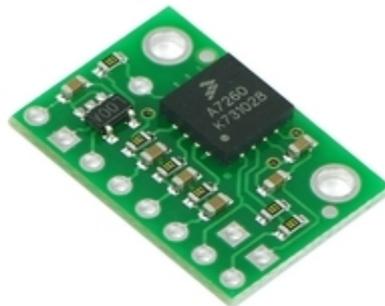


Abb 5. MMA7260QT Sensor [Freescale 08: MMA7260QT]

The carrier board addresses both issues while keeping the overall size as compact as possible.

The datasheet will be found by clicking the following link: [MMA7260QT datasheet](#)

Functionality

It can be modelled as a set of beams attached to a movable central mass that move between fixed beams. The movable beams can be deflected from their rest position by subjecting the system to an acceleration. As the beams attached to the central mass move, the distance from them to the fixed beams on one side will increase by the same amount that the distance to the fixed beams on the other side decreases. The change in distance is a measure of acceleration. The g-cell beams form two back-to-back capacitors. As the center beam moves with acceleration, the distance between the beams changes and each capacitor's value will change, ($C = A\epsilon/D$). Where A is the area of the beam, ϵ is the dielectric constant, and D is the distance between the beams. The ASIC uses switched capacitor

techniques to measure the g-cell capacitors and extract the acceleration data from the difference between the two capacitors. The ASIC also signal conditions and filters (switched capacitor) the signal, providing a high level output voltage that is ratiometric and proportional to acceleration.

Programming

To measure with the MMA you have to use the AD-converter and the multiplexer to choose between 3 channels if you want to measure the acceleration in every direction. After converting one axis don't forget to switch to the next channel. You have to set the reference voltage on 5V too. After the A/D conversion is done it is recommended to create a function to convert the result to mg (milli g) The output can be realized by using the UART in connection with the Hyperterminal.

Miscellaneous

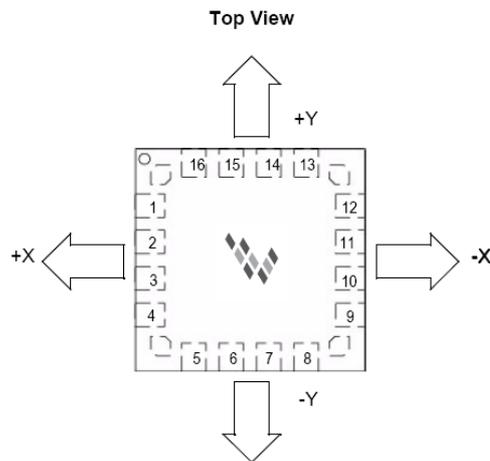


Abb 6. Axis-Allocation [Freescale 08: MMA7260QT]

- the arrows in the picture above indicates direction of the mass movement, up and down is the z-axis.
- function to calculate the output to mg $\frac{\text{result of axis} \cdot 100}{41} - 204$

(try to create your own function it is not guaranteed that it works everywhere)
- use delays between the the conversions to make sure that the AD-converter works correctly

2.3.2 Memsic 2125 Dual-Axis

Introduction

The Memsic 2125 is a sensor for measuring the static and dynamic acceleration with a range of ± 2 g for X- & Y-Axis. Due, it's an thermal accelerometer, there is also the possibility to measuring the actual temperature of this sensor.

With a microcontroller, like the ATmega168, the following things can be done:

- Dual-axis tilt sensing for autonomous robotics applications (BOE-Bot, Toddler, SumoBot)
- Single-axis rotational position sensing
- Movement/Lack-of-movement sensing for alarm systems

Functionality

- working voltage is 5V at a low power consumption of 4 mA
- one analogue output for measurement of temperature
- two digital outputs for acceleration, each X- & Y-Axis

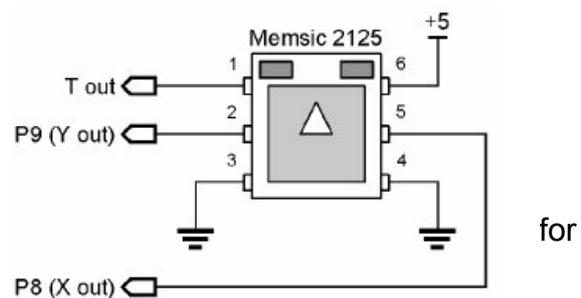
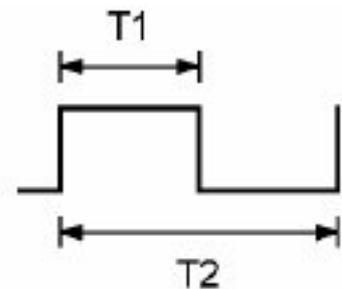


Abb 7. Memsic 2125 [Paralax 08: Memsic 2125]

Due to this outputs the sensor can be directly connected to the Arduino microcontroller board, only one analogue and two digital inputs are necessary (don't forget the 5V and GND).

The digital outputs for X- & Y-axis are given as periodical pulses. One period, called T2, has the total duration of 10 milliseconds. The measured value of acceleration will be given to you by the pulse width of the high signal T1. At 0 g the Memsic 2125 is calibrated to give out a pulse which width is the half of the period (5 ms).



For calculating the g-force the following formula is given: $A(g) = ((T2/T1) - 0,5) / 12,5\%$

There is a small heater inside the sensor, to heat a little bubble of air (for further information take a look to the datasheet), so the temperature can also be measured. The pin “T out” can also directly be connected to the ADC (analogue to digital converter) and is calibrated to stay at a voltage level of 1,25 V at 25 °C. For every degree the temperature changes, there is a voltage change of 0,005V (5 mV).

Programming

The main task of the program is the measurement of the pulse width T1. There are different ways of programming to get nearly the same result, because the Atmel ATmega168 has much powerful functions to process with PWM-signals (pulse width modulated). But for better understanding we chose the easier way of polling the high signal.

Programming steps to interpret the g-force:

- declare the digital Port/PINX as Input
- detect the rising edge of the high signal by polling
- if it's detected, start the internal timer/counter of the microcontroller
- detect the falling edge and stop the timer/counter immediately
- the timer register TCNT1 returns the time the high level will hold in system clocks
- calculating the g-force and output it via the UART
- don't forget to clear the timer register before the next count starts

To get the measurements for X- and Y-axis is in these two cases the same, only the Port/Pin is different.

Programming steps to get the temperature:

- initialize the ADC and select the right channel for input
- start the AD-converting and wait till it has finished it's work
- read the 10-bit value that stands for the voltage the sensor gives
- calculating and output

Miscellaneous

G-force measurement:

- for the timer/counter there is a division factor available, because if you don't use it the timer register will be overflowed (16 millions quartzclocks * max. pulse length of 10 milliseconds = 160K, it doesn't fit into 16bit), so we used the division by 8 (only CS11 bit is set and starts also the timer)
- for readout the timer register it's important to read at first the low-byte then the hi-byte, the datasheet of the ATmega168 says so
- for calculating the right values of g-force we used a little changed formula: pre-calc the counter value to real microseconds via counter value *0,5 (16 MHz divided by scaling factor 8 = 2 MHz, 1/this = time in seconds, converted to microseconds a general factor of 0,5 resulted, means one shift to the left) then put it to the formula: $mg = ((T1/10) - 500) * 8$, so you will get an signed integer value of milli-g

Temperature measurement:

- we used the whole 10bit value, because 1 bit stands for a change of one degree Celsius (full range of 5V divided by changing value of 0,005V = 1000 steps, equal to the 10bit ADC of 1024 steps), makes the calculating very easy: only the offset of 231 has to be subtracted (231 = value for 1,25 V at 25 °C)
- you can interpolate the output by taking more measurements, adding the result to the pre-result and after that, divide it by the count of measurements

2.3.3 HM55B Compass Module

Introduction

The Hitachi HM55B Compass Module is a dual-axis magnetic field sensor. The module is controlled by an serial interface and has an own communication protocol. Therefore it is harder to use than an analogue sensor, but the protocol is quite “straight forward”. It is not too hard to implement it for the Arduino-board.

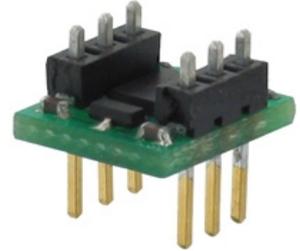


Abb 8. HM55B Compass Module [Paralax 08: HM55B]

Features

- The Sensitivity amounts $1\mu\text{T}$ for magnetic field strength
- The range of measurement is between $-180\mu\text{T}$ and $180\mu\text{T}$
- resolving magnetic field measurements into two component axes
- It takes just 40 ms from “start of measurement” to “result is ready”
- Built-in resistor protection for data pins eliminates bus conflict risks
- Compact and breadboard-friendly 0.3 inch, 6-pin DIP package

For further information you can download the complete data sheet of this sensor here: <http://www.parallax.com/Portals/0/Downloads/docs/prod/compshop/HM55BModDocs.pdf>

Functionality

Even if this sensor is called “compass” it is “just” an sensor for magnetic field strength. It won't give you the direction to north directly, but the sensor has two axes, so you can easily calculate the direction out of the two measurements.

$$\text{angle}(\theta) = \arctan(-y/x)$$

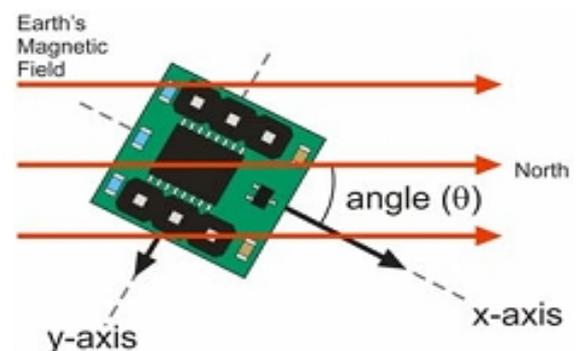


Abb 9. Magnetic Field Sensor [Paralax 08: HM55B]

2.3.3 HM55B Compass Module

The resulting value is an angle between -180° and 180° . 0° means the sensor points to north. If the result is $\pm 180^\circ$ the module points to south.

The sensor generates two 11bit values in “two's complement”. Therefore it is necessary to pad a 16bit value with ones, if the determined value is negative. (It is negative if the 11th bit is “high”). (See also http://en.wikipedia.org/wiki/Two's_complement)

Programming

There are two different kinds of interfacing the HM55B Module. We decided to use the 4-wire interface, because it seems to be better understandable. The following chapter tries to explain the communication protocol.

The general principle is to shift out 4bit commands to the sensor and to receive the results from the module. All values transmitted to and received from the module are most significant bit first, with the bit value valid after the clock signal's rising edge.

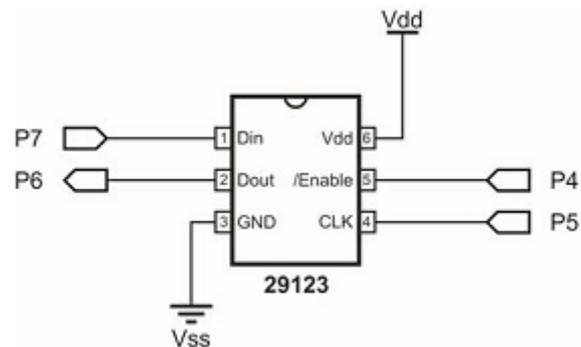


Abb 10. Pin-allocation [Paralax 08: HM55B]

To reset the HM55B, set /EN from high to low, and shift-out “0000”, then set /EN high again.

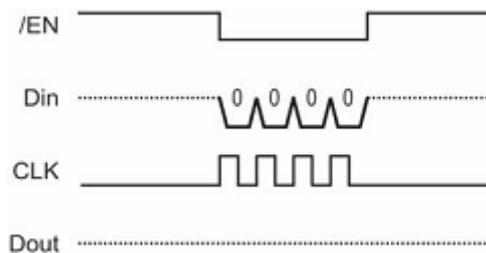


Abb 11. Reset of the HM55B [Paralax 08: HM55B]

The following table shows all possible commands shifted out to the module:

Binary Value	Meaning
„0000“	Reset device
„0001“	Start measurement
„0011“	Report measurement status (and transmit the result if it's ready)

2.3.3 HM55B Compass Module

The Compass Module will reply to the report measurement status command with one of these values:

Binary Value	Meaning
„AABB“	The first bits (“AA”) indicate measurement completion (“end flag”), the second bit (“BB”) indicate measurement errors (error flag)
„1100“	„11“ → measurement completed; „00“ → no errors
„00XX“	Measurement still in progress, or device has been reset“
„XX11“	/EN did not receive low-high-low impulse signal between start and report commands

After reset, start a measurement by taking /EN low again, then shift-out “1000”. **Leave /EN “low” until checking the measurement status.**

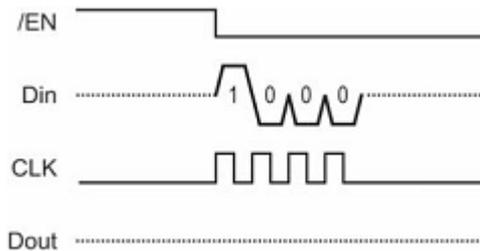


Abb 12. Start of measurement [Paralax 08: HM55B]

To check the measurement status, start by sending a positive impulse to /EN. Then, shift-out “1100”, and shift-in the status flags. While the measurement is in progress, the end flag and error flag will both be “00”. The compass Module may be polled for status repeatedly until the measurement is complete, at which point the end flag will change to 11. Alternatively you can just implement a delay of 40ms. The measurement will be completed afterwards. (Don't forget: You also have to send an “1100” after the 40ms, to the result of the measurement.)

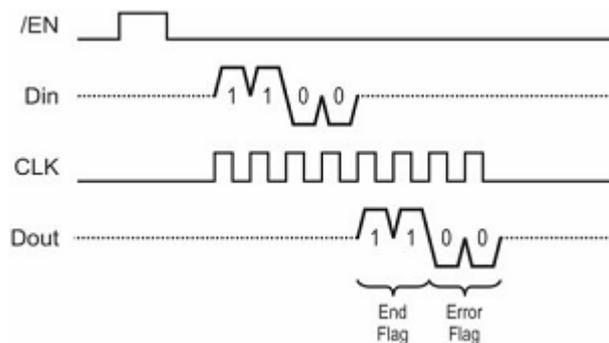


Abb 13. Status of measurement [Paralax 08: HM55B]

Upon receipt of “1100”, discontinue polling. Leave /EN “low”, and move on to shifting-in the x and y-axis values.

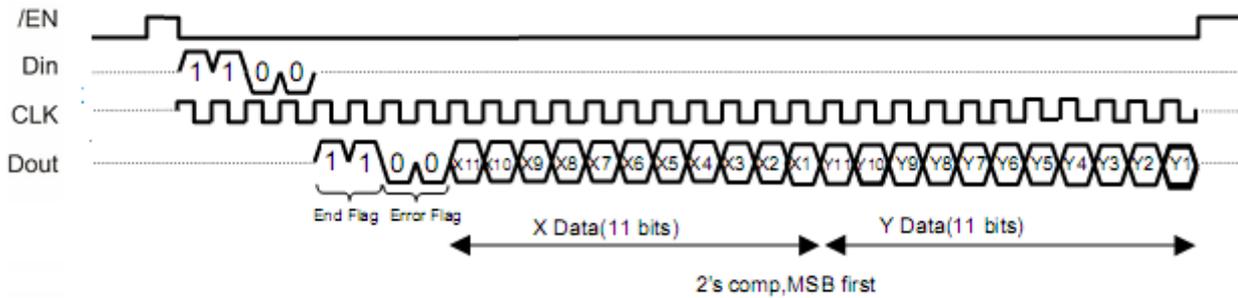


Abb 14. Result of measurement [Paralax 08: HM55B]

After getting both values set /EN “high” again.

Precautions

- Do not apply voltages to the device that are outside the values stated in the Pin Definitions and Ratings section.
- Do not operate or store the Compass Module near sources of strong magnetic fields. Strong magnetic fields can be created by bar and ring magnets, electric motors, and other coil elements such as solenoids, relays, and large inductors.
- Do not apply magnetic fields in excess of 300 μT to the Compass Module. Magnetic fields stronger than 300 μT can permanently damage the sensor.
- Mount the Compass Module as far away as possible from magnetic field disturbances. These include magnets (including compass needles), motors, power cords, coils, metal boxes, and sometimes the ground.

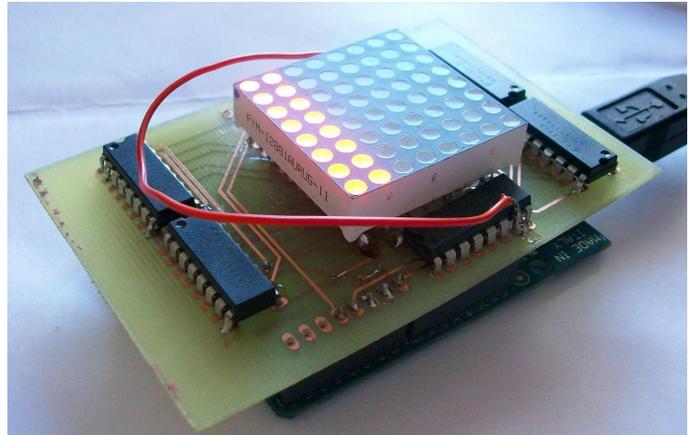
Miscellaneous

On the one hand it is possible to send an output via Hyperterminal (See also “Output via Hyperterminal”). On the other hand it is much more fun to do it with the LED-matrix on the Display-Board. This is quite hard to program, because you have to control the LED-matrix via interrupts and simultaneously you have to ask the sensor for next values via polling. At the same time you also have to generate an output-image, out of the determined values. It's may be a harder task, but it works fine with the Arduino-Board.

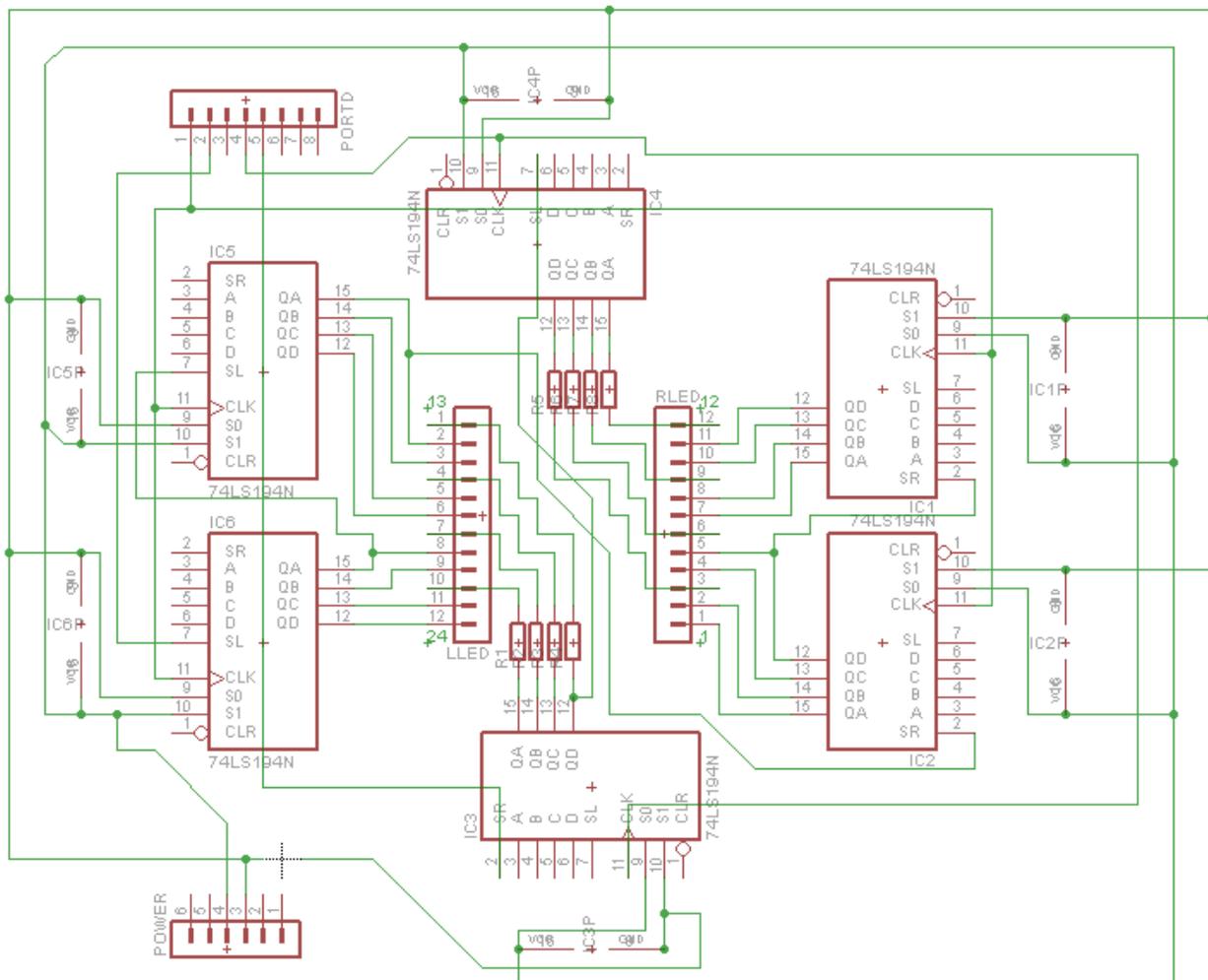
3 Display-Board

3.1 Introduction / Description

The „Display Board“ contains 6 of a 4-Bit Shiftregister and a 8x8 LED-Matrix. The LED-Matrix is able to show two colours, red and green. With the 24 pins on the backside you can control the colour and the position of the active LEDs.



3.2 PIN-Allocation / Circuit Diagram



IC1, IC2, IC5, IC6 : 4 x 4 Bit Shiftregister
IC3, IC4: 2 x 4 Bit Shiftregister
LLED,RLED: Connectors to the LED-Matrix
POWER: Powerconnector
PORTD: CLK and DATA connectors

8 Pins are for the horizontal red LEDs, 8 for the horizontal green LEDs and 8 Pins for the vertical line. The pins are connected to the 6 shiftregisters.

As shown in following table, the SV4 board is the PDD port of the Arduino board.

Port	used for
PD7	CLK_X
PD6	DATA_X
PD4	CLK_Y
PD3	DATA_Y

The first register contains 4 of the 6 Shiftregister, so it can hold 16BIT (WORD).

The first bit of the word is for the first red LED, the second for the first green.

In this way, the LED1 to LED4 is used by the the first BYTE, the last ones by the second BYTE.

The second register contains the other 2 shiftregister for the vertical line.

To activate a line there must be a LOW in the register. (LOW-active).

3.3 Programming

To use the shiftregisters, there must be feeded a Data-BIT on the DATA-input and a HIGH edge to the CLK.

After that it is necessary to put the CLK on LOW again.

Now the next BIT is on turn. In this way, it is possible to shift a whole NIBBLE (4 BIT) into the shiftregister.

For example, to activate only the red LED of the first line, there must be a 7Fh (01111111b) in the y-register and an AAAA (10101010 10101010) in the x-register.

The way to get the data in the registers, is to set the DATA pin to the value of the first bit and a rising edge to the CLK pin. After the CLK the content of the register is shifted and the LSB is set to the value of the DATA pin. The same operation should be done for the next bits, until the whole value is shifted into the register.

If the time between the HIGH active CLK is very long, it is possible to see the movements of the Bits (in the registers) by switching the LEDs on and off.

To multiplex, it is recommend to put in the X data very fast. After 16 HIGH risings on CLK_X the complete WORD is loaded. If it is done fast enough, you couldn't see the 16 shift operations. Then you have to wait a short moment of time before setting a rising edge to CLK_Y to go on with the next line. This steps should be repeated continuously.

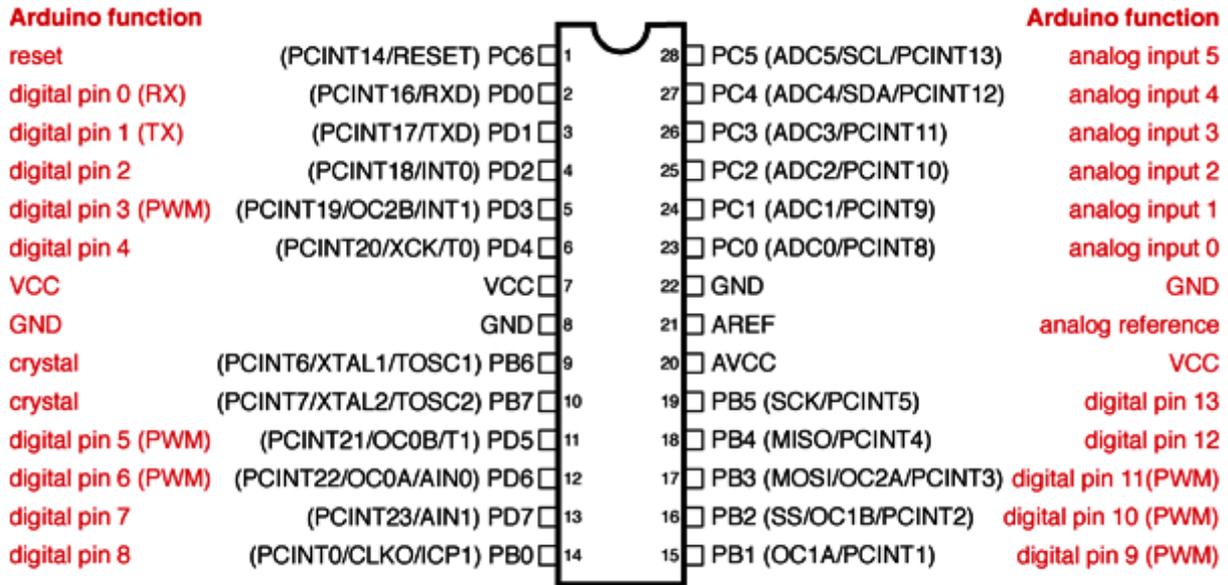
With exact timings, it seems that the 8x8 picture is shown on the matrix.

3.4 Miscellaneous

- It is possible to use an interrupt to drive the LED matrix.
- With a timing controlled interrupt about 1ms and with eight lines, the LEDs are changing by 125Hz.
- If you activate the red and the green light at the same time, its look like an orange light

4 Helpful Information

4.1 Arduino Pin Mapping



4.2 Output via Hyperterminal

To get out the measured values from the microcontroller, the internal serial interface is very useful. With the aid of ATmega’s Universal Synchronous and Asynchronous serial Receiver and Transmitter (USART) it’s very easy to connect the microcontroller to the PC or other devices for controlling and processing. For all sensors we used an asynchronous 9-bit transmitting mode to output the measured values and displayed it with MS Hyperterminal (an easy communication tool, part of the MS Windows XP distribution) on the PC.

Due to the most modern PC’s (mainly the mobile ones) are build without a classic serial port, the Arduino is equipped with an USB port and a special IC (called FTDI) to convert the signals from RS232 to USB. So in the first step it’s necessary to install a USB-to-COM Driver on the PC, this is a COM-Port emulating.

Next step is to initialize the serial interface on the ATmega168. The standard mode is set to asynchronous 9-bit transmission, so there will be nothing to do. Then the right Baud rate is to choose. This is the value for the speed the bits will be transmitted. In our case we chose 9600 Baud. With the formula: 16 MHz clock divided by (16*9600 Baud) and finally subtracted

4.2 Output via Hyperterminal

by one we get the right counter value for the internal Baud clock generator. This value, 103 decimal, we have to write it in the UBRR0 register (16-bit). To be able to send something, the transmitter enable bit (TXEN0 in UCSR0B-reg) has to be set.

On receiving site the Hyperterminal has to be configuring correctly: select the right COM-Port, set the Baud rate to 9600, 8 data bits and one for parity.

In the programs we declared two functions to realize the communication:

```
int uart_putc(unsigned char c){      // UART-Output for one char only

    while(!(UCSR0A & (1 << UDRE0))) {
    }
    UDR0 = c;
    return 0;
}

void uart_puts (char *s)           // output of a char-string
                                   // takes the uart_putc for each character
    while (*s){
        uart_putc(*s);
        s++;
    }
}
```

The user calls the `uart_puts()`-subroutine by a value consists of a string of char (one dimensional text array). This subroutine called the other ones (`uart_putc()`) and give it one character after another. The `uart_putc` subroutine writes the given character into the shared transmit-receive data register UBR0 and wait till the transmitter has finished it's work. The received character will be displayed immediately in MS Hyperterminal communication window.

There is one problem left: if you want to transmit numbers (values from the sensors f.e.) they have to be converted into text before. For this work there are special functions given in the standard library that comes with AVRStudio, so bound it with “`#include <stdlib.h>`”.

The following commands will convert integer values into text strings:

- `utoa (x , text , 10);` converts the unsigned integer variable x to the string text of char, the 10 stands for the decimal system of numbers
- `itoa (x , text , 10);` the same as above but for signed integer values

To get a Linefeed & Carriage Return we will send finally a “`\n\r`”.

For more complex communication operations (in both direction i.e.) it will be very useful to use the interrupt system of the ATmega168. But only for transmitting it's the easiest way like we did !

II. Bibliography

[Parallax 08: PIR] PIR Sensor:
<http://www.parallax.com/Portals/0/Downloads/docs/prod/audiovis/PIRSensor-V1.2.pdf>

[Parallax 08: PING] PING))):
<http://www.parallax.com/Portals/0/Downloads/docs/prod/acc/28015-PING-v1.5.pdf>

[Sharp-World 08: GP2Y0A021YK] GP2Y0A021YK:
http://sharp-world.com/products/device/lineup/data/pdf/datasheet/gp2y0a21yk_e.pdf

[Freescale 08: MMA7260QT] MMA7260QT:
http://www.freescale.com/files/sensors/doc/data_sheet/MMA7260QT.pdf

[Parallax 08: Memsic 2125] Memsic 2125 Dual-axis Acc:
<http://www.parallax.com/dl/docs/prod/acc/memsickit.pdf>

[Parallax 08: HM55B] HM55B:
<http://www.parallax.com/dl/docs/prod/compshop/HM55BModDocs.pdf>