



Merlin Systems Corp. Ltd

**Miabot BT PRO
Magnetic Compass Sensor**

(prototype)

v0.1

Revision History

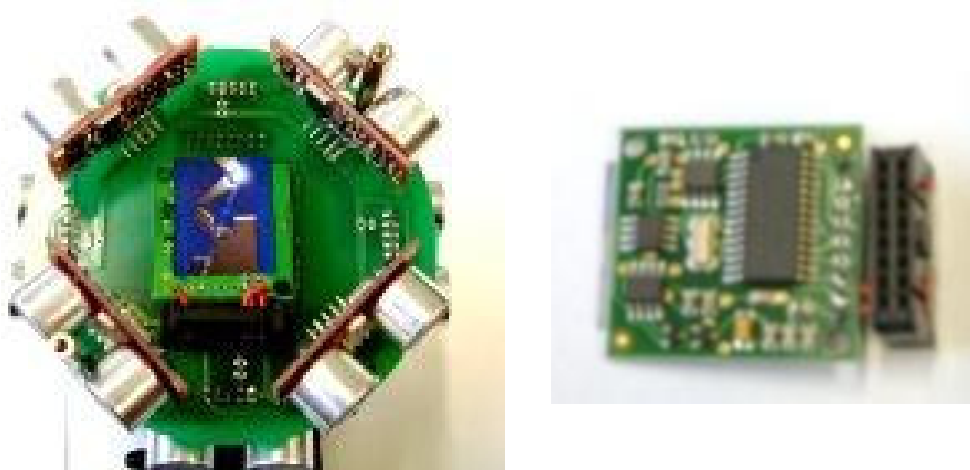
Vo.1 25/11/05 pp prototype version

© Merlin Systems Corp. Ltd 2002-2004

Merlin Systems Corp. Ltd assumes no responsibility for any errors which may appear in this manual, reserves the right to alter the devices, software or specifications detailed herein at any time without notice, and does not make any commitment to update the information contained herein. Merlin Systems Corp. Ltd's products are not authorized for use as critical components in life support devices or systems.

Introduction

The Miabot Pro magnetic compass is a small sensor PCB that fits onto the top of the robot and plugs into an expansion port.



The compass connects to the robot I²C bus, and can be interrogated using the standard I2C control commands (as also used for Sonar and Video Camera accessories). A small PC test application is also provided.

NOTE: The compass module can only be used with Miabot BT PRO robots, software v2.3 and above.

Compass Board Hardware

The compass device is based on the Devantech type CMPS03 module.

This has internal calibration operation, and a potential accuracy of around 3°.

However, on the Miabot robots the operation of the compass tends to be compromised by the magnetic field of the drive motors, and the achievable accuracy is generally less.

See the separate manufacturers datasheet for more information on the basic device (“CMPS03 documentation.htm”, on the support disk).

The compass connects via the I²C bus (robot commands to access the I²C bus are described in a separate section below).

All compasses have a *fixed* I²C address of 0xC0 (or upper 7 bits thereof), so only one compass can be fitted to a robot.

Fitting

The prototype board is wired to a 2×8-way IDC connector, which plugs directly into the Pro expansion connector.

Mechanically, this can be fitted direct to the Pro robot expansion (top) board, or on the top of a Sonar Array board (see picture).

The first-version compass boards are *not* physically compatible with the “Gripper / Expansion-IO board” connectors. However, this is expected to change in future versions.

NOTES:

The compass module *must* be kept horizontal for sensible direction readings.

In the current arrangement, the compass sensor is actually upside-down, so this must be taken into account (actual bearing = 360°-reading).

The compass operation is *considerably* affected by the magnetic fields coming from the drive motors (permanent magnets).

The VB test application provides an example of how the readings can be corrected for the static field –see below.

Proposed Future Changes

Because of the magnitude of the motor magnetic field, it is generally unsuitable to mount the compass directly onto the Pro main board: This is so close to the motors that very poor results can be expected.

In a future version, the compass will be made mountable via a flexible lead to avoid this problem.

In addition, a future firmware upgrade will provide a specialised compass-reading command that can apply a calibrated correction stored in the robot EEPROM.

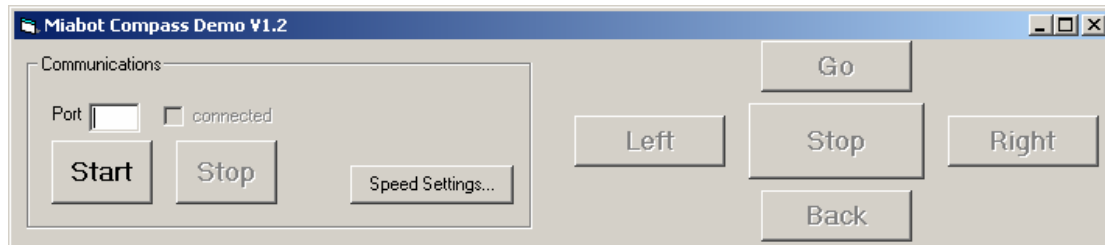
(i.e. similar correction calculation to that shown in the demonstration application, but embedded in the robot itself)

Demonstration Software

A simple example Visual Basic application is provided, which can be installed from the Support Disk ("Miabot Compass 1v2.exe").

The source code is also provided on the disk.

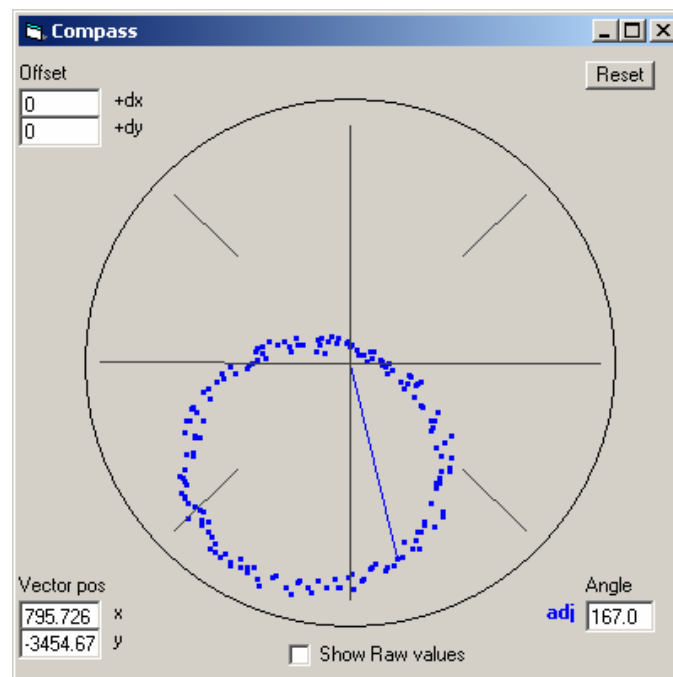
When run, this presents a window like this :-



Type the Bluetooth virtual Comport number into the 'Port' box, and hit the 'Start' button to start communication with the robot.

(The 'Stop' button on the left will stop communications and close the port).

At this point, the application will pop up a compass display window, like this :-



The compass screen shows the module output as a vector (blue line), and also a kind of 'data history' as the set of blue dots.

The buttons on the right can now also be used to steer the robot around.

(The "Speed Settings" button also brings up controls to change the movement speeds)

It is useful to exercise the compass module by hitting the 'Left' or 'Right' buttons to make the robot rotate constantly.

The 'history' dots in the above example show a rough circle from rotating the robot slowly around (see below for movement controls).

The circle is offset from the centre, showing the fixed field from the robot motors (which rotates with the compass, of course).

In practice, the data will not usually look as tidy as this example. The internals of the compass sensor appear to have a variable gain factor, and nearby magnetisable objects can also considerably distort the picture. These both produce significant nonlinear artifacts.

Controls

Other elements also appear on the compass screen :–

“Vector pos” – x and y

show the current raw sensor output values from the magnetic sensors
editing these has no effect

“Angle”

shows the current angle to the X,Y position defined by the module output, which is our estimate of the magnetic north bearing.

“Reset” button

controls automatic saving of history and display scaling: The display rescales automatically to accomodate the largest data. Both history and scaling are reset when this is pressed.

“Offset” – x and y

these control/display the correction offset values (see below)

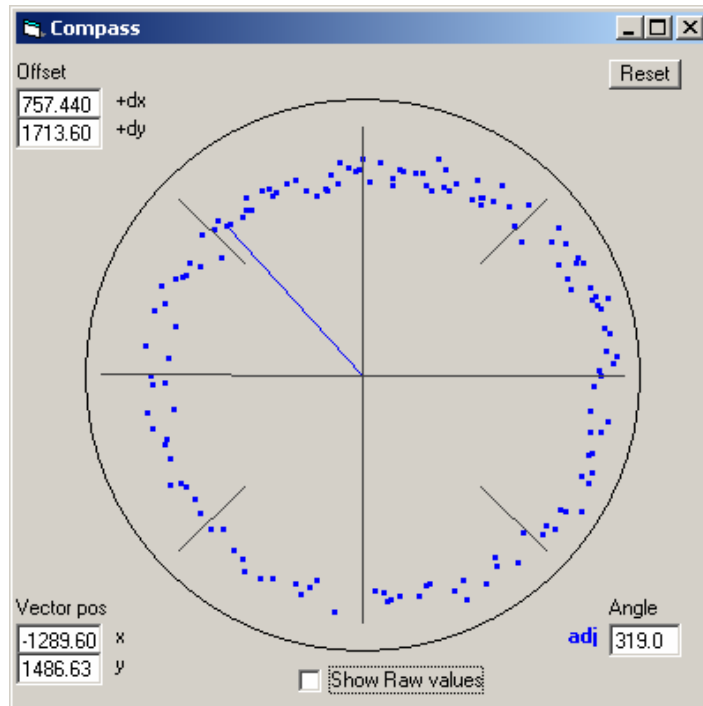
“Show raw values”

shows both raw and corrected history values and angles on the screen (see below)

Calibration and Corrections

By clicking within the compass screen, the “Offset” x and y values are set to adjust the received values so that the point ‘clicked on’ becomes the centre of the display.

The history is flushed, but if the robot is rotating the history will eventually rebuild itself. The display might now look like this, for example :-



The intended effect of this is to enable you to ‘centre’ the pattern so that angle readings are more representative of the true magnetic bearing.

The effect is to approximately ‘subtract’ the fixed magnetic field vector due to the motors.

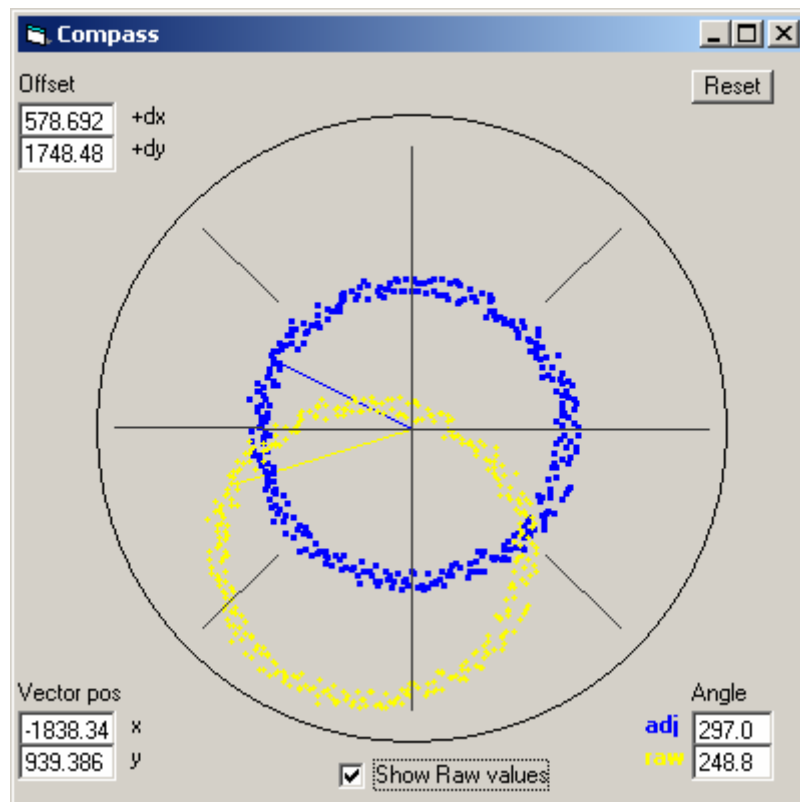
The size and direction of this field vary between individual robots. The magnitude is also highly sensitive to the surrounding hardware and the height above the main pcb that the compass is mounted.

NOTE: This can only correct for an offset field fixed relative to the compass. Other errors, such as introduced by nearby metalwork or fixed magnetic objects, can not be removed in this way.

The pattern can be re-centred at any time by clicking again at the new desired ‘centre’ point.

The same effect can be produced by adjusting the “Offset” ‘x’ and ‘y’ values in the text boxes. (However, it is then best to also hit ‘Reset’).

The “Show raw values” tickbox also allows you to display raw and processed data side-by-side, like this :-



This show clearly the effect that the offset adjustment has on the angle calculation.

Miabot Pro Commands for I²C Support

From version 2.3 onward, the Miabot Pro command-set contains a command to control slave devices on the the I²C bus, as follows :-

[i]-access I²C bus device

Two command formats allow reading and writing over the I²C bus :

[iAA=x1,x2...]

write data bytes (hexadecimal values x1,x2 etc.) to the bus at address 'AA'.

[iAA?nn]

read nn (decimal) bytes from the bus at address 'AA'.

The bus addresses 'AA' are hexadecimal values, of which the top 7 bits are the actual I²C address and the lowest bit must always be 0 (i.e. even numbers only).

Accesses with no data transferred are permitted - i.e. read or write 'no bytes'.

These are specified as **[iAA=]** and **[iAA?]**.

If an error occurs (including no bus response), the access will be retried a set number of times. If the maximum retries have occurred without a response, the command fails with an error code. (See below for retry controls and error codes).

All 'i' commands eventually produce a response string, but this may not occur for some time if a lengthy retry is specified. The following are the possible response formats :-

<i>

a write access succeeded (N.B. also read access with no bytes)

<i x1 x2 ...>

a read access returned the (hexadecimal) bytes shown (x1, x2...)

<i=nn>

the access failed, with error-code nn

Current error-code values are -

<i=1> access timed out: no response after N retries (see below)

<i=2> access rejected: expected acknowledge bit missing (during write only)

All I²C accesses are automatically retried, according to two parameter settings -

[.iN] number of bus-access retries -can be zero. Default = 3

[.iT] interval between bus-retries (in milliseconds)

Examples:

[iC4?3]	read three bytes from I ² C address 0x62	-->	response <i 25 A7 03>
[iA0=1,21]	write two bytes to address 0x50	-->	response <i>
[i70?14]	read 14 bytes from 0x38 (no device)	-->	response <i=1> (failed)

Commands to Read Compass

To read results from the compass, two accesses are required each time (see datasheet for full explanation) :-

- (1) First, send a register address
e.g. “[iC0=2]”
- (2) Then, read out data starting at that address
e.g. “[iC0?4]” (to read back the first 5 echoes as 5* byte-pairs)

The arrangement of data values in the registers is explained in the data sheet.

For our purposes, it is best to read out the raw sensor values, which are at addresses 4 to 7. For example –

“[iC0=4]” with (eventual) response “<i>”
 “[iC0?4]” with (eventual) response “<i: 01 25 FE 1B>”

The results are MSB-first, so here –

- the X-value is hex 0125 = +293 decimal,
- the Y-value is hex FE1B = –485 decimal (16 bit 2's complement)