Gearo 1 Motor Controller

User's Manual Rev1.0

V1.0

Jul 2019

1. INTRODUCTION

This document describes the intended use of the Sigenics Gearo1 brushless DC motor controller. The Gearo1 controller includes an on-board microcontroller with factory-loaded firmware for closed-loop PID control of the motor shaft angle, and high-level control via i2c serial digital bus.

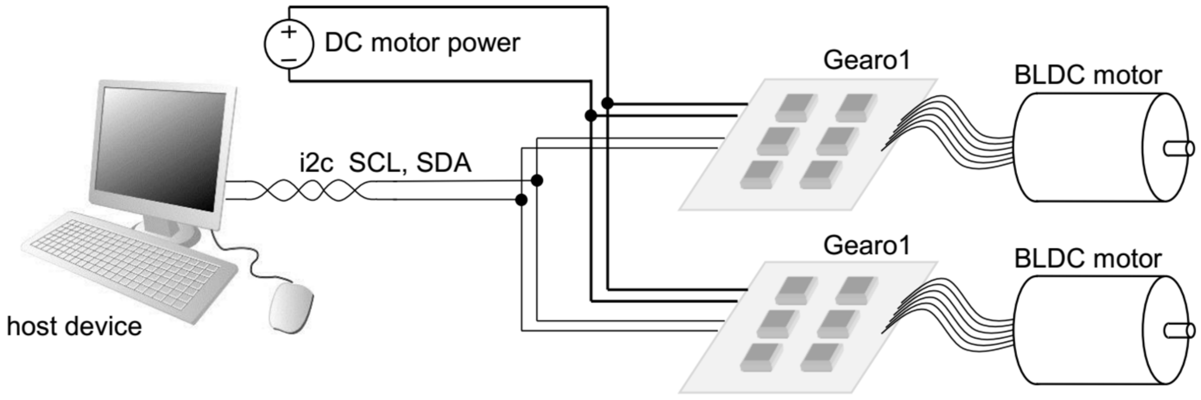


Figure 1.1 Illustration of the Gearo1 in its intended usage context. The host device sends i2c commands to a number of Gearo1 units on the bus, and each Gearo1 unit controls one motor.
A single host can control multiple motors on a 4-wire bus.
host device illustration credit: © Can Stock Photo / kgtoh

The host device may be any device with an i2c master port, such as a single-board computer (e.g., BeagleBone or Raspberry Pi), or any of a variety of microcontrollers, or a conventional computer with appropriate i2c bus adapter.

The Gearo1 has a 7-bit i2c bus address, with address 0x00 reserved for “broadcast to all” messages, so a single i1c bus may communicate with as many as 127 Gearo1 units. The factory-default Gearo1 i2c address is 0x2A. The i2c bus address of a Gearo1 device can be changed by an i2c command from the host device.

Each Gearo1 unit comprises a PCBA with microcontroller, local power regulation, and power switching circuitry. The Gearo1 microcontroller contains a firmware program which implements the local PID motor control, and manages communications between the Gearo1 unit and the host via the i2c port.

1.1. Gearo1 Hardware

The Gearo1 hardware consists of a microcontroller, local power regulation for the microcontroller, and a 3-phase H bridge driver.

The associated demo kit may contain additional hardware components such as a motor, cables, and so on.

1.2. Gearo1 Firmware

The Gearo1 factory-loaded firmware includes functions for i2c communication, motor phase detection, motor winding drive, and PID control.

The firmware communication function receives commands and issues status information through the i2c port.

The motor phase detection function controls the motor winding drive angle according to the state of 3 input terminals intended for connection to the BLDC motor's hall sensors.

The motor winding drive function generates a PWM drive control output to the 3-phase H bridge, based on information from the motor phase detection and PID control functions.

The PID control function adjusts the motor drive power and direction according to the difference between a set-point received from the communication function, versus the actual motor position as detected by the motor phase detection function. PID coefficients can be adjusted by i2c commands.

The control firmware and command set are detailed in the later sections of this user guide document.

2. DEFINITIONS

2.1. Gearo1 New Sigenics device which is specified in this document, also referenced by its Sigenics internal part number SG1MC-BL-12V

2.2. BLDC Brushless DC – this is a type of motor which has no mechanical commutators for the various motor windings; commutation of the motor windings is managed using electronic control

2.3. PCBA Printed circuit board assembly

2.4. PID Proportional, Integral, Derivative – this is a common category of second-order linear feedback control

2.5. PWM Pulse Width Modulation – this is a method for efficiently delivering a variable level of electrical power to a load device

2.6. PV Process Variable – this is the variable controlled by the internal PID control loop programmed into the Gearo1 device

2.7. SP SetPoint – this is the target value for the process variable

3. MECHANICAL INFORMATION

3.1. Electrical connection ports

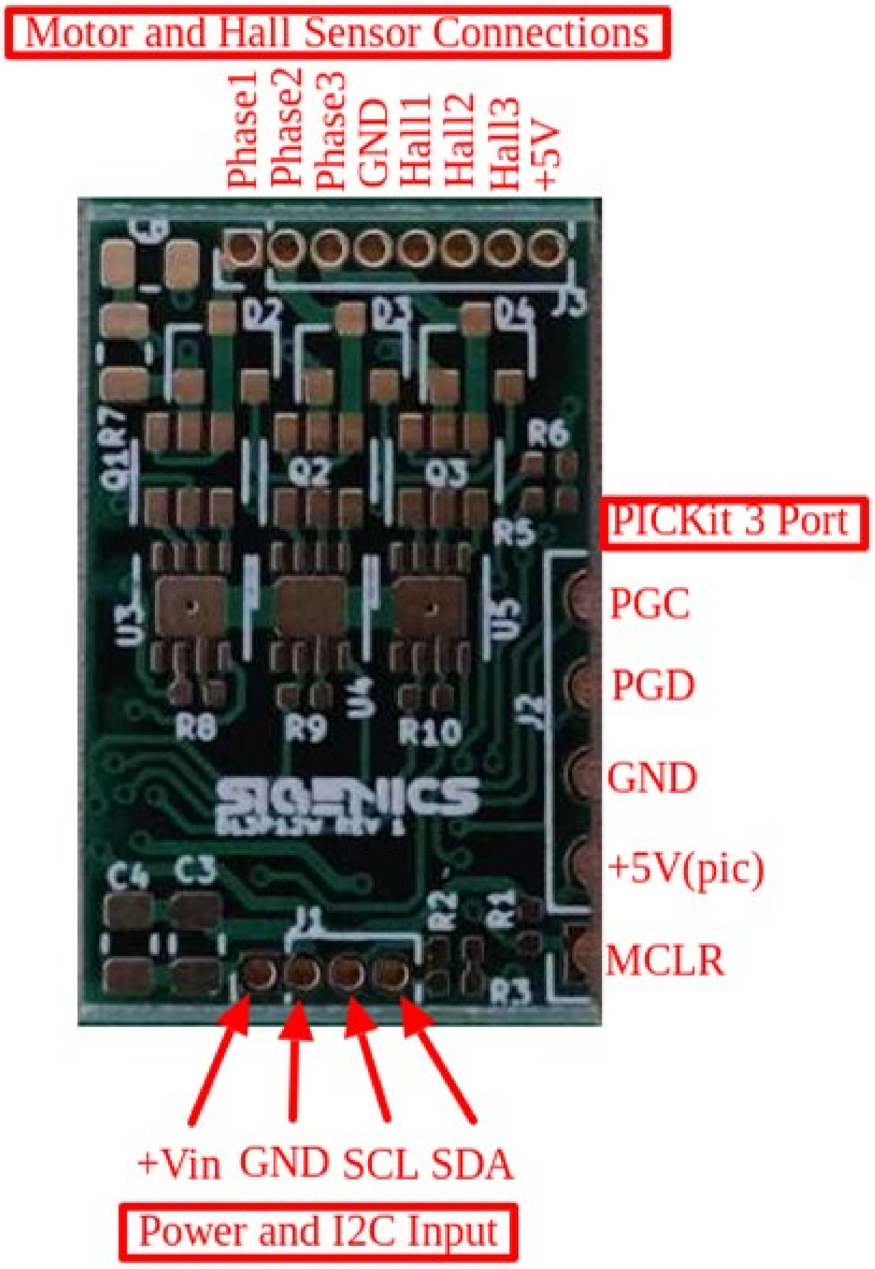
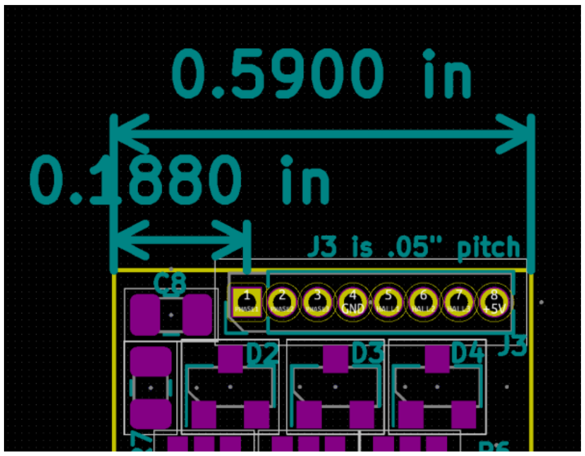


Figure 3.2 Illustration of the Gearo1 connection terminals. The power and i2c input connection allows connection to a shared bus. The motor and hall sensor terminals are for connections to the motor that the Gearo1 unit controls. The PICKit3 port is for optional uploading of alternative firmware.

3.2. Size

The Gearo1 board has a total mass of 1.9 grams, and is a small rectangular assembly as illustrated below. The board has components soldered to both sides, and is 0.59 inches x 0.91 inches x 0.07 inches (15mm x 23.2mm x 1.6mm).



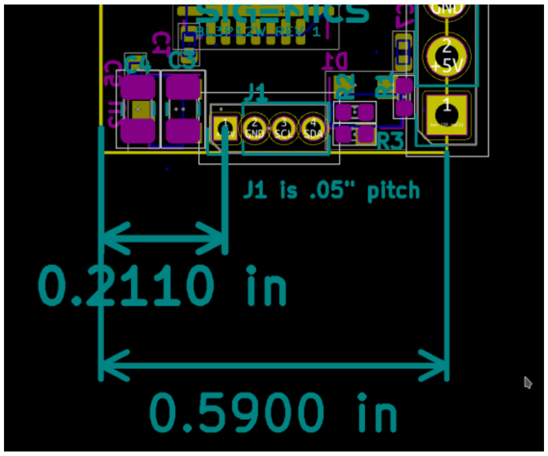


Figure 3.1 Illustration of the Gearo1 board dimensions. Note that the PICKit3 connector is a set of castellated holes along the edge of the board, with a pitch (center-to-center spacing) of 0.10 inches (2.54mm). The pitch of the bus connector is 0.05 inches (1.27mm). The pitch of the motor connector is 0.05 inches (1.27mm). The "pin 1" location of each connector is marked with a square pad.

4. SPECIFICATIONS

4.1. Electrical

Electrical specification items are grouped by category.

	Specification	*Description*
6.2.1.	**DC Power Supply**	
6.2.1.1.	DC supply voltage	12V nominal (7V minimum, 20V maximum)
6.2.1.2.	Quiescent current draw (motor off, supply at 12V)	Needs to be determined by bench test
6.2.2.	**i2c interface**	
6.2.2.1.	Address of 3PP device	Programmable 7-bit address. General call address to all units at 0x00. Reserved addresses are: 0x1-0x7, 0x78-0x7F
6.2.2.2.	SCL clock pulse width	5us minimum (standard low-speed 100kbps)
6.2.2.3.	on-board 3PP pull-up resistor on SCL	10k ohms +/- 5%
6.2.2.4.	on-board 3PP pull-up resistor on SDA	10k ohms +/- 5%

6.2.2.5.	on-board 3PP digital power supply for pull-up on SCL, SDA	5.0V +/- 5%
6.2.3.	**motor phase sense**	
6.2.3.1.	power supply output to Hall effect sensors	5.0V +/- 5%
6.2.3.2.	number of Hall sensor inputs to the 3PP board	3
6.2.4.	**motor winding output**	
6.2.4.1.	maximum output current at 50% duty cycle, 12V power supply	1.5A (this needs to be verified by bench test)
6.2.4.2.	PWM frequency	Adjustable from 10KHz to 100kHz
6.2.4.3.	PWM adjustment precision	8 bit
6.2.4.4.	PWM duty cycle range	0% to 50% (v0.1 firmware)

4.2. Mechanical

	Specification	*Description*
6.3.1.	Dimensions	15mm x 23.2mm x 1.6mm
6.3.2.	Mass	Less than 2g

4.3. Environmental

	Specification	*Description*
6.4.1.	Storage temperature range	0C < T < 70C
6.4.2.	Operating temperature range	15C < T < 45C

5. HOST COMMANDS

Commands are communicated from the host device to the Gearo1 device over the i2c wires as newline terminated, ASCII-encoded strings. The Gearo1 device, when used with factory-installed firmware, acts as a “slave” device on the i2c bus.

Each command is preceded by one byte indicating the 7-bit slave unit address and the R/W status of the command. For more details on the i2c bus protocol, a wide variety of readily available reference sources are available (for example, https://en.wikipedia.org/wiki/I%C2%B2C).

[Command Character] [Argument Character(s)] [Newline ('\n') Character]

For example, to write a new setpoint of -1000 to the device, the following string would be sent to the Gearo1:

“S-1000\n”

In this example, the command character is “S”, and “-1000” are the argument characters, and there is a newline character to terminate the message. Other commands have no argument, for example to set Idle Mode requires only a command character and newline:

“X\n”

Arguments must be no longer than 14 characters. The table below lists the commands, and the proper format of their respective arguments.

Command	Character	Description
Set Idle	X	Sets the operational mode to Idle Mode.
Set Active	G	Sets the operational mode to Active Mode.
Write Setpoint	S [arg]	Writes a new setpoint. [arg] is an integer value between -32768 and 32767.
Zero PV	Z	Sets the process variable to 0.
Write P	P [arg]	Writes a new value to the Kp PID coefficient. [arg] must be an ASCII-encoded string representing a valid floating point value i.e. “0.0035”, “-0.57”, “69483.6”, etc. The Kp value is stored in non-volatile memory, and persists through power cycle events.
Write I	I [arg]	Writes a new value to the Ki PID coefficient. [arg] must be an ASCII-encoded string representing a valid floating point value i.e. “0.0035”, “-0.57”, “69483.6”, etc. The Ki value is stored in non-volatile memory, and persists through power cycle events.
Write D	D [arg]	Writes a new value to the Kd PID coefficient. [arg] must be an ASCII-encoded string representing a valid floating point value i.e. “0.0035”, “-0.57”, “69483.6”, etc. The Kd value is stored in non-volatile memory, and persists through power cycle events.
Write Bias	B [arg]	Writes a new value to the bias, which is added directly to the output of the PID calculation. [arg] must be an ASCII-encoded string representing a valid floating point value i.e. “0.0035”, “-0.57”, “69483.6”, etc. The bias value is stored in non-volatile memory, and persists through power cycle events.
Set PWM frequency	F [arg]	Sets the PWM frequency (argument is in units of cycles per second).
Motor active	A	Turns on the 3-phase bridge PWM drive output.
Motor idle	I	Disables the 3-phase bridge PWM drive output.
Set i2c address	T [arg]	Sets the i2c address to the 7b value in the LSB portion of arg.
Save User Settings	V	Saves all of the PID constants, PWM frequency, and I2C address into nonvolatile memory.

All commands and arguments are ASCII-encoded character strings. All command arguments (where applicable) take the form of ASCII-encoded decimal numbers, floating point in the case of the PID coefficients, and signed 32-bit integers in the case of the setpoint (fractional values will be truncated).

To read back the 32-bit process variable, simply issue a standard i2c read of 4 bytes. The process variable will be issued most-significant byte first in **binary encoded** 32-bit signed integer format.

An I2C read command can be issued at any time (or not). Any number of bytes can be read, it is not necessary to read them all, for example if you are only interested in the setpoint you can read back only 4 bytes. In chronological order, the following will be read back during an i2c read operation:

setpoint most-significant byte (including sign bit)

setpoint 2nd most-significant byte

setpoint 3rd most-significant byte

setpoint least-significant byte

status flag byte

The status flag byte is intended to contain information about the various kinds of errors that might occur. These things include: invalid command/parameter, invalid hall sensor value (0 or 7 are not legit hall outputs and indicate an error), overcurrent events, and possibly other useful information. If all of the bits are zero, everything is ok. The following bits will be set in the case of an error:

bit 0: invalid command/parameter

bit 1: input buffer overflow

bit 2: overcurrent event

bit 3: math/conversion error

bit 4: brownout reset detected

bit 5: hall sensor error

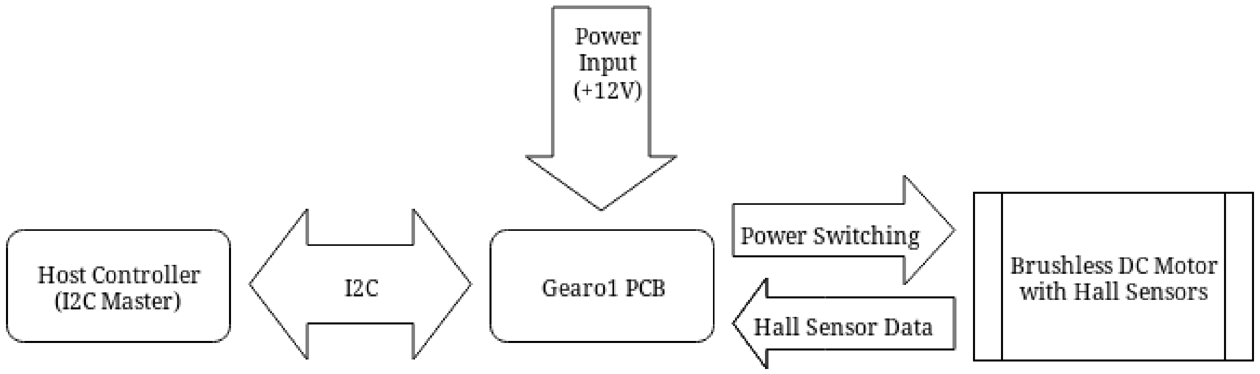
bit 6/7: reserved for future use

Following a read transaction, the bits of the status flag byte are all cleared by the Gearo1 V1.0 firmware.

6. Gearo1 V1.0 Firmware operation

This section describes the operation of the firmware in the Gearo1 controller board as shipped from Sigenics. Note that the Gearo1 includes a programmable microcontroller device, so the operational characteristics can be altered by the user if desired.

The Gearo1 board is a small circuit designed to drive a small 3-phase brushless DC motor. The following diagram shows a logical overview of the system.


Power Input (+12V)
Host Controller (I2C Master)
I2C
Gearo1 PCB
Power Switching
Hall Sensor Data
Brushless DC Motor with Hall Sensors

The Grearo1 firmware implements a PID control loop to control the position of the motor. The firmware command set allows the user to modify the PID coefficients, write a new setpoint, zero the process variable, and other similar tasks. The process variable corresponds to the position of the motor as indicated by the motor's hall sensor outputs.

The process variable (i.e. motor position count) is implemented as a 16-bit signed integer. Under the default demonstrative firmware configuration (v1.0) the process variable defaults to 0 on startup regardless of motor shaft position.

The device firmware currently included in existing Grearo1 devices (v1.0) is a beta release which is configured for basic demonstration purposes without any specific application or implementation in mind.

Commands are issued to the Grearo1 device using the i2c protocol. All commands take the form of a newline terminated, ASCII-encoded string of characters. The first character of the string determines the command, while the remaining characters (not including the terminating newline character) comprise the argument (where applicable).

6.1. Operational modes

6.1.1. Idle Mode

In this Idle Mode, the PID algorithm is disengaged, and the drive transistors are off. The process variable (aka motor shaft position) continues to be tracked normally. This mode is useful for writing multiple changes to the PID variables and having them all take effect simultaneously by re-engaging active mode.

6.1.2. Active Mode

In Active Mode, the PID algorithm is engaged and the drive transistors are engaged as necessary to cause the process variable to track the setpoint.

6.2. Factory defaults

As shipped from Sigenics, the Grearo1 device includes firmware with the characteristics described in this document, an i2c bus address of 0x2A, and factory default PID coefficients of

- Proportional coefficient Kp = 0.15
- Integral coefficient Ki = 0.02
- Derivative coefficient Kd = 0

- Constant Bias = 0

User commands can alter these factory default settings, as described above in Section 5. The PID coefficients and device bus address are stored in non-volatile memory, so this information will remain stable through power supply brownout or power cycle events.

6.3. Startup

On startup, the Grearo1 device with v1.0 firmware defaults to Idle Mode, with a setpoint (SP) set to 0 and process variable (PV) of 0. The startup condition occurs when the Gearo1 device is first powered up, or when a hard reset event occurs (this may happen during a power supply brownout).

The startup default values for the PID variables and coefficients are as loaded from nonvolatile memory on startup. If the user has previously saved user settings using the "Save User Settings" command (V), then these values will be loaded on startup. Otherwise the factory defaults will be loaded as per 6.2.

6.4. PID control loop behavior

Here we include a description of the PID loop calculations (define the inputs and outputs, and the calculations that go from input to output)

The Gearo1 controller includes a PID control loop programmed into the firmware of its on-board microcontroller. This control loop has inputs and outputs of:

- Input: process variable setpoint from the host control interface
- Input: motor shaft angle from the motor's hall sensors
- Output: motor drive vector to the 3 motor windings
- Output: duty cycle of PWM to motor windings

The v1.0 firmware uses the shaft angle as the process variable. The shaft angle is encoded in units of hall sensor ticks. The motor drive vector magnitude is encoded as PWM duty cycle, and the drive vector direction is encoded as a signed selection of the motor windings to be driven on any given PWM cycle.

In the v1.0 firmware, the motor drive vector direction is quantized according to a lookup table, and the motor drive magnitude is calculated according to a PID time step calculation. Hall sensor inputs are φ1, φ2, φ3, and motor winding outputs are w1, w2, w3. The v0.1 drive vector direction lookup table is:

Hall sensor input				PID drive sign						
				forward				reverse		
φ1	φ2	φ3		w1	w2	w3		w1	w2	w3
0	0	1		P	O	N		N	O	P
1	0	1		P	N	O		N	P	O
1	0	0		O	N	P		O	P	N
1	1	0		N	O	P		P	O	N
0	1	0		N	P	O		P	N	O
0	1	1		O	P	N		O	N	P

P = motor winding terminal is connected to the positive side of the drive bridge
N = motor winding terminal is connected to the negative side of the drive bridge
O = motor winding terminal is open (not driven)

The table above is ordered top to bottom as a forward shaft rotation sequence.

The PID control loop in v1.0 firmware calculates an updated PWM continually, based on the error between the setpoint and process variable measured at the start of the calculation. The process variable PV is the motor shaft angle, in units of hall sensor ticks (60 degree motor shaft increments). NOTE that the motor supplied in the demo kit includes a 139:1 gear ratio reduction from motor shaft angle to output shaft angle. Updates to PWM magnitude are calculated as:

```
dt = time interval since previous update
PV +=  change from previous shaft angle if applicable (either forward or reverse)
E = (SP – PV)                * calculate the present value of control error
Int += (E * dt)              * calculate an update of the integrated error
Der = ((E – Eprev)/dt)       * calculate the derivative error
PWM = (Ki * Int) + (Kp * E) + (Kd * Der) + Bias
```

The PWM duty cycle is constrained, so if the above calculated PWM result exceeds a hard-coded max value of 75%, the updated PWM value is set to 75%, and if the calculated PWM result is less than 0, the updated PWM value is set to 0%. The PWM result is scaled and truncated proportionally to the value in the time-base register for the PWM module (CCP1PRL), and is then written to CCPR1B.

NOTE1: Internal numerical values used in the PID loop calculation are represented in the v1.0 firmware as 32b floating point values. Numerical values commanded by the host device are sent over the i2c bus as ASCII strings, e.g., “379.6” and interpreted into 32b floating point values.

NOTE2: The Hall sensor input sequence for a “forward” shaft rotation can differ substantially among the various different motor manufacturers and may even differ between different motor types from a single manufacturer. So, the v1.0 drive vector table may produce inappropriate results for motors other than those supplied in the demo kit.

NOTE3: The process variable is internally represented as a 32b signed integer, with no special provision for overflow or underflow detection. So, if the motor shaft is programmed to a setpoint close to the maximum or minimum 32b signed integer value, and an external torque forces the shaft angle further (causing the PV variable to roll over), the PID loop will respond as though the shaft angle was suddenly at a large negative value, and therefore attempt to drive the shaft angle a long way forward. Rollover handling can be implemented in firmware, but as the correct response would be driven by the needs of the application, this is currently unimplemented.

NOTE4: Like the process variable, the error is also currently implemented as a 32-bit signed integer. This means that although the total possible error magnitude in the process variable is 4294967295 (-2147483648 to 2147483647) , the maximum magnitude possible to fit in the error variable is 2147483647 (or -2147483648 in the negative direction). Therefore, error values of greater than 2147483647 will currently result in anomalous behavior. Fortunately, 2147483647 max error still gives plenty of room for an adequate demonstration, and clever use of the “Zero PV” command can allow the motor to spin beyond the range of the max error magnitude or process variable. Error overflow handling can be implemented in firmware, but as the correct response would be driven by the needs of the application, this is currently unimplemented.

7. USAGE NOTES

Here we have an assortment of notes regarding practical issues when using the Gearo1 device.

Possible issues to mention are:

7.1. Power supply source impedance

To work properly, the Gearo1 device must have an adequate power supply.

If the power supply source impedance is too large, then high-current transients (which may occur when the motor loading changes) can cause various problems. The total power supply impedance seen at the Gearo1 terminals includes the native impedance of the power supply itself, plus the resistance and inductance of the cabling which connects from the power supply to the actual terminals of the Gearo1 board.

One possible problem with too large a source impedance on the power supply and cabling is a brownout condition which may reset the control processor.

Another possible problem, especially with cable and connector impedance, is that large motor currents may shift the ground level at the Gearo1 terminals enough to cause problems with the i2c signaling (see the next section for more discussion of possible i2c signaling issues).

Generally the best solution is to use the largest practical wire gauge, and the shortest practical wire length, for the power supply distribution.

7.2. I2C signaling

To work properly, the i2c communication bus must be properly connected, with care for issues like signal pull-up, ground current interference, and termination of long cables.

7.2.1. I2c pull-up current

Each Gearo1 unit includes a pair of 10kΩ resistors pulling up the SCL and SDA lines of the i2c bus. If a user’s application includes a sizable number of Gearo1 units sharing a single i2c bus, the total pull-up current may become too large for reliable signaling. The solution to this issue is to request Gearo1 boards without pull-up resistors, or, the user can remove resistors R2 and R3 from several of the Gearo1 units, as needed, to limit the total pull-up current.

7.2.2. Ground current

The i2c bus of the Gearo1 is referred to the GND terminal, which is also the return current pathway for the motor drive power. This shared connection opens the possibility that the motor drive current may degrade the i2c signal integrity.