What is SmartDrive?

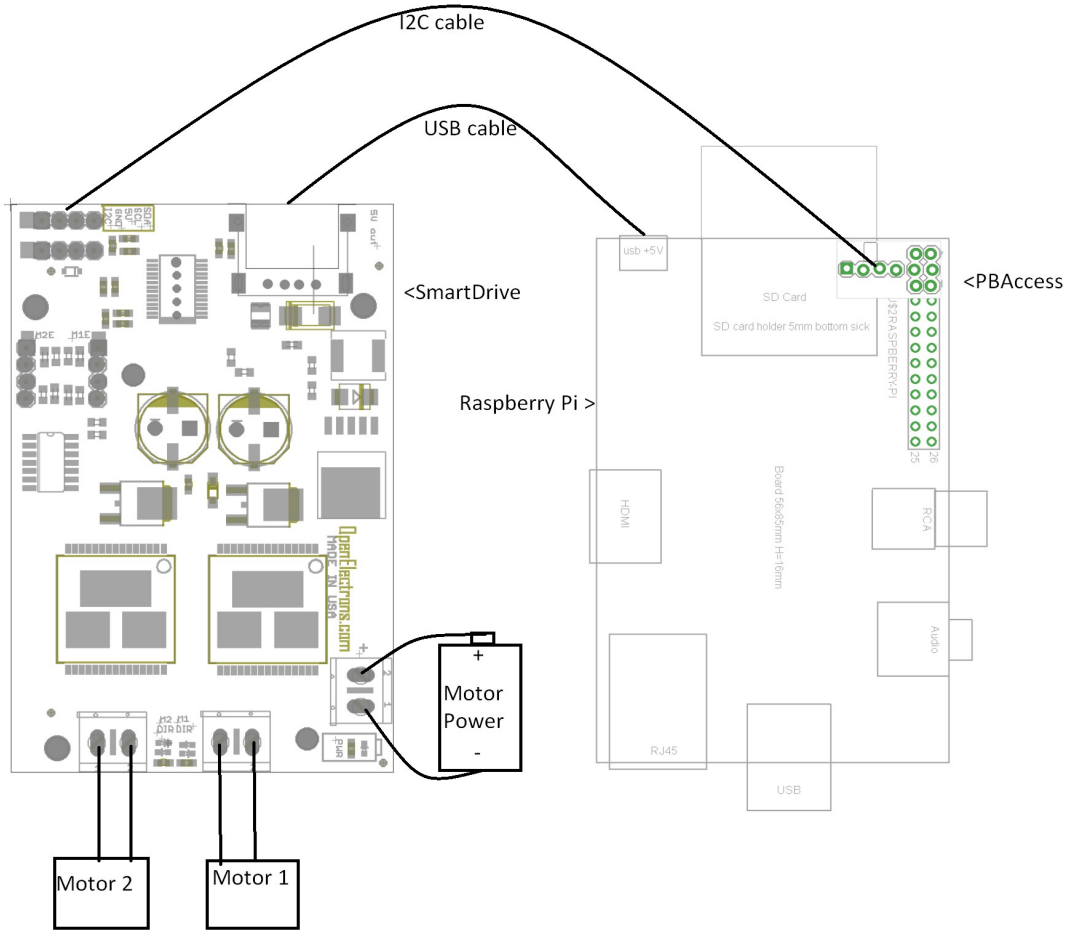
SmartDrive is a multiplexer to control high current DC motors with the Raspberry Pi.

Connections and Placement

SmartDrive can be connected to I2C Access Point board to either of the I2C header pins at the top right corner of the device. Match pin marked by white box (GND) on SmartDrive to pin with white box on I2C Access Point board.

There is a USB connector to on the top left corner of the device used to **power the Raspberry Pi**.

Two motors can be connected to SmartDrive through the two green connectors on the bottom left of the board labeled M1 and M2.



If using PID control, the encoders are to be connected to the the corresponding 4 pin header located on the left side of the device also labeled M1 and M2.

Encoder Pinout:

M1E	M2E
VCC	VCC
A	A
GND	GND
B	B

⚠ **NOTE**

Ensure to check the pinout of your encoder in its data sheet or user guide for proper connection.

Input power is to be connected to the green connector located on the bottom left of the device.

⚠ **NOTE**

I2C header pins only support I2C or Digital sensors or multiplexers. If you intend to daisy chain several SmartDrives or other digital sensors, please read the relevant section from the Advanced Information Appendix of this doc.

Mounting SmartDrive on your design

There are five 1/8in. diameter holes on the on the SmartDrive that are used for mounting the device to your Raspberry Pi. The holes are located on the left (under the I2C pins), middle (just to the right of the M1 encoder header pins), right (just beside the USB port), and two on the bottom of the board.

To mount SmartDrive on your Raspberry Pi, line up the middle and bottom right with holes on Raspberry Pi. Use a nylon or plastic standoff between the two boards.

⚠ **NOTE**

DO NOT use a metal standoff. Boards will short which will prevent proper function and may damage the devices.

Supplying power to your SmartDrive

The SmartDrive has a Green terminal on the bottom right of the board to connect external battery.

⚠ **NOTE**

SmartDrive is rated for external power supply of 24 volts DC max. Do not exceed 16 volts DC for reliable output. Ensure to not exceed this value. While connecting external battery, ensure correct polarity.

Battery Options:
12V battery (recommended).

RC rechargeable battery:
9.6-13 Volts Ni-CD or Ni-MH RC rechargeable battery.

You can buy such battery and it's charger in local toy store.

To connect these batteries to SmartDrive, you can purchase following connector cable from mindsensors.com website: http://www.mindsensors.com/rpi/10-rc-battery-pack-connector-cable

Lithium-Polymer Battery (also known as Li-Po or LiPo batteries)
With SmartDrive, use 9.6-13 Volts battery. Select the capacity based on your load and anticipated duration of use.

You can purchase these batteries in electronic hobby stores, or on-line RC toys stores. An example url is here:

Feature Highlights

Table below lists the important features provided by the SmartDrive.

Feature	Description
Timed Control of each motor	Each motor can be run for a specified duration of time.
Encoder control of each motor	Each motor can be run from its current Encoder position to a new position (with or without a specific speed).
Speed control of each motor	Speed of each motor can be controlled in timed run or encoder based run.
Brake Vs Float while	Each motor can be set to Brake (where

Feature	Description
stopping the motors	motor shaft can not be turned easily), Vs Float (where motor shaft is free to rotate by external force.
Holding Encoder position	At the end of run, hold the encoder position (i.e. motor turned by external force is restored to last set encoder position).
Turning motor by degrees	Move it in forward or reverse direction.
Turning motor by rotations	360 degrees makes one rotation.
Running operations asynchronously	While a motor is running other operations may be performed.
Running motors for unlimited duration.	While motors are running, you can also perform other operations. NOTE: When motors are set to run for 'Unlimited Duration', they will continue to run until a Stop command is issued (or power is disconnected). In other words, after starting the motors for 'Unlimited Duration' if your program exits without stopping the motors, they will continue to run.
Stopping motors abruptly.	
Reading Motor Encoders	You can read the value of each encoder from the SmartDrive.

Programming Techniques for SmartDrive

Python Method:

SmartDrive was designed for use with Raspberry Pi. All APIs are written and python.

Download SmartDrive and dependent libraries directly to your Raspberry Pi from pip using: **sudo pip install SmartDrive**

Online SmartDrive Programming Reference can be found at http://openelectrons.com/OEPiReference/html/class_smart_drive_1_1_smart_drive.html

SmartDrive programming examples can be found at http://www.mindsensors.com/index.php?controller=attachment&id_attachment=123

Other Methods:

SmartDrive can be used with other controllers using programming methods not supported by mindsensors.com. To make this possible you must write your own library for the desired method.

Current Characteristics

Average current consumption of this device is about 16.0 mA.

I2C Bus address

Factory Default Address: 0x36

Changing the I2C Bus Address:
Address change instructions can be found at
http://www.mindsensors.com/content/54-how-to-change-i2c-address-using-raspberry-pi

APPENDIX A - Advanced Information

I2C Registers:

The SmartDrive appears as a set of registers as follows:

Register	Read	Write
0x00-0x07	Firmware version – *Vxxxx*	-
0x08-0x0f	Vendor Id – *Openelec*	-
0x10-0x17	Device ID – SmartDrive	-
0x41	You can read the SmartDrive battery voltage at this register. (voltage in milli-volts = register value * ??).	I2C Command
	Motor 1 Write Parameters	
0x42	Encoder Target for Motor 1 (long) 0x42: Least Significant Byte 0x43: Byte 2 0x44: Byte 3 0x45: Most Significant Byte	Encoder Target of Motor 1 (long)
0x46	Speed for Motor 1 (byte)	Speed for Motor 1 (byte)
0x47	Time to run in seconds for Motor 1 (byte)	Time to run in seconds for Motor 1 (byte)
0x48	Command register B for Motor 1	Command register B for Motor 1 (set this value to 0 as this is for future use)
0x49	Command register A for Motor 1 (read the description below for details of this register).	Command register A for Motor 1 (read the description below for details of this register).
	Motor 2 Write Parameters	
0x4A	Encoder target for Motor 2 (long) 0x4A: Least Significant Byte 0x4B: Byte 2 0x4C: Byte 3 0x4D: Most Significant Byte	Encoder Value of Motor 2 (long)
0x4E	Speed for Motor 2 (byte)	Speed for Motor 2 (byte)
0x4F	Time to run in seconds for	Time to run in seconds for

	Motor 2 (byte)	Motor 2 (byte)
0x50	Command register B for Motor 2	Command register B for Motor 2
0x51	Command register A for Motor 2 (read the description below for details of this register).	Command register A for Motor 2 (read the description below for details of this register).
	Motor Read Parameters	
0x52	Encoder position of Motor 1 (long) 0x52: Least Significant Byte 0x53: Byte 2 0x54: Byte 3 0x55: Most Significant byte	-
0x56	Encoder position of Motor 2 (long) 0x56: Least Significant Byte 0x57: Byte 2 0x58: Byte 3 0x59: Most Significant Byte	-
0x5A	Status Motor 1 (byte). See section below for details of this register.	
0x5B	Status Motor 2 (byte). See section below for details of this register.	
0x5C	Tasks Running for Motor 1 (byte)	
0x5D	Tasks Running for Motor 2 (byte)	
	Registers for Advanced PID control	**Writing these registers has immediate effect on operation. These registers will be reset to factory default values upon power cycle.**
0x5E	Kp for Encoder Position Control (int) 0x5E: Least Significant Byte 0x5F: Most Significant Byte	Kp for Encoder Position Control (int)
0x60	Ki for Encoder Position Control (int) 0x60: Least Significant Byte 0x61: Most Significant Byte	Ki for Encoder Position Control (int)
0x62	Kd for Encoder Position Control (int) 0x62: Least Significant Byte	Kd for Encoder Position Control (int)

	0x63: Most Significant Byte	
0x64	Kp for Speed Control (int) 0x64: Least Significant Byte 0x65: Most Significant Byte	Kp for Speed Control (int)
0x66	Ki for Speed Control (int) 0x66: Least Significant Byte 0x67: Most Significant Byte	Ki for Speed Control (int)
0x68	Kd for Speed Control (int) 0x68: Least Significant Byte 0x69: Most Significant Byte	Kd for Speed Control (int)
0x6A	Pass Count – The PID controller repeatedly reads internal encoder ticks, this is the number of times the encoder ticks reading should be within tolerance. (default 5)	Pass Count – Higher Pass count gives more time to position internal encoder, thus providing better accuracy in positioning, but will take longer time. Change this only if you need different motor positioning speeds and accuracy. (For normal usage, default values will be OK).
0x6B	Tolerance – The Tolerance (in ticks) for encoder positioning. (default 80).	Tolerance – the accuracy you desire while positioning. Low number will position the encoders more accurately, but may take longer time. Change this only if you need different motor positioning speeds and accuracy. (For normal usage, default values will be OK).
	Power Data Registers	**Power information of the SmartDrive**
0x6E	Battery voltage	Battery voltage value displayed in millivolts.
0x6F	Reset status	
0x70	Motor 1 current value	Current at Motor 1 – for future use, not yet implemented.
0x72	Motor 2 current value	Current at Motor 2 – for future use, not yet implemented.

Supported I2C Commands:

CMD	Hex	Description
R	0x52	Reset all Encoder values and motor parameters. (This does not reset the PID parameters).
S	0x53	Issue commands to both motors at the same time, for

CMD	Hex	Description
		Synchronized starting of both motors.
		Motor Stopping Commands
a	0x61	Motor 1: Float while stopping.
b	0x62	Motor 2: Float while stopping.
c	0x63	Motor 1 and 2: Float while stopping.
A	0x41	Motor 1: Brake while stopping.
B	0x42	Motor 2: Brake while stopping.
C	0x43	Motor 1 & 2: Brake while stopping.
		Encoder Reset Commands
H	0x72	Motor 1: Reset Encoder to zero
I	0x73	Motor 2: Reset Encoder to zero

These commands are issued on command register (0x41).

Motor Command Register Explained

Each motor has two command registers (Register A and Register B). In current release Register B is reserved for future use and must be set to zero.

Bits in Register A should be set to 1 to avail functionality as described below.

Register Bit	Turn this bit to 1 for following functionality
Least significant bit (bit 0)	**Speed control of your motor.** The SmartDrive will honor speed values specified in the speed register for the respective motor.
Bit 1	**Ramp the speed up or down.** While starting the motor or changing speed, the SmartDrive will ramp up or ramp down the power to new value. If this bit is zero, the power changes are sudden, e.g. full power is applied to motors as they start.
Bit 2	**Relative change based on encoder values.** This is useful when Bit 3 is turned on, and in this case, the SmartDrive will make a relative movement from last seen Encoder position (it will add the new Encoder position to old position and move to that resulting position). Useful when turning by degrees or rotations. If this bit is 0, the Encoder positions are taken as absolute values.
Bit 3	**Encoder control of your motor.** The SmartDrive will honor Encoder values specified in Encoder position register for respective motor. If speed values are also specified, and speed bit is set on, motors will rotate to new encoder position with the specified speed.

Register Bit	Turn this bit to 1 for following functionality
Bit 4	**Brake or Float at the completion of motor movement.** If this bit is 1, motor will Brake at the completion, otherwise it will float.
Bit 5	**Encoder active feedback.** This bit is used when Encoder control is used. If this bit is set to 1 at the completion of motor movement, the SmartDrive will continue to hold the Encoder position (i.e. if motor is turned by external force, SmartDrive will try to restore it to last specified Encoder position). If this bit is zero, the motor may float.
Bit 6	**Timed control of your motor.** The SmartDrive will honor the time specified value in 'Time to run' register and run the motor for specified time. (If Time control bit as well as Encoder Control bit is on, the Timed control has precedence over encoder control).
Most significant bit (bit 7)	***GO.*** When this bit is set to 1, all above bit values are brought into effect. This is useful to synchronized starting of both motors. As it takes some time to write each motor's register values. The I2C command 83 can be used to change these bits to 1 for both motors at once.

Motor Status Register Explained

Each motor has one status register. Each bit in status register indicates various situations with the motor as explained below.

Register Bit	Value 1 indicates the situation is true
Least significant bit (bit 0)	**Speed Control is ON.** Motor is programmed to move at a fixed speed.
Bit 1	**Motor is Ramping (up or down).** If the Power ramp is enabled, this bit is 1 while the motor is ramping (while changing its speed).
Bit 2	**Motor is powered.** (This may not mean motor is moving.)
Bit 3	**Positional Control is ON.** The motor is either moving towards desired encoder position or holding its position.
Bit 4	**Motor is in Brake mode.** (0 value of this bit means motor is floating).
Bit 5	**Motor is overloaded.** If the external load prevents motor from achieving desired speed, this bit is set to 1.
Bit 6	**Motor is in timed mode.** This bit is 1 while the motor

Register Bit	Value 1 indicates the situation is true
	is programmed to move for given duration.
Most significant bit (bit 7)	**Motor is stalled.** The external load caused the motor to stop moving.

Running Motors for Unlimited Duration

Not specifying Encoder Control or Timed Control bit will result in unlimited running of motor, (the speed control is honored if specified). To stop motors started with 'Unlimited Duration' use respective Stop command from the command set.

⚠ **NOTE**

When motors are set to run for 'Unlimited Duration', they will continue to run until a Stop command is issued (or power is disconnected). In other words, after starting the motors for 'Unlimited Duration' if your program exits without stopping the motors, they will continue to run.

How to detect if motor is moving or not moving:

In General:

Anytime when the 'Stalled' bit is 1, the motor is not moving.

Running in encoder mode:

During moving:

Position Control bit (bit 3) is 1
'Motor is Powered' bit (bit 2) is 1

Finished moving normally:

All bits are zero

Stopped due to stall:

Position Control bit (bit 3) is 1
'Motor is Powered' bit (bit 2) is 1
Stalled bit (bit 7) is 1

Running in timed mode:

During moving:

Timed Mode bit (bit 6) is 1
'Motor is Powered' bit (bit 2) is 1

Finished moving normally:

All bits are zero

Stopped due to stall (while time is not over):

Timed Mode bit (bit 6) is 1
'Motor is Powered' bit (bit 2) is 1
Stalled bit (bit 7) is 1

Stopped due to stall (after time is over):

All bits are zero.

Upgrading SmartDrive firmware:

SmartDrive firmware is upgradeable through the Raspberry Pi. Firmware upgrade instructions can be found at http://www.mindsensors.com/content/37-how-to-upgrade-firmware-using-pi