An Intelligent Winch for Vertical Profiling

Most instruments used for environmental observations in natural settings such as oceans, estuaries, lakes, and reservoirs measure at a single point in space. Commonly used single point sensors measure properties of water such as temperature, salinity, turbidity, pH oxygen and chlorophyll content. Properties such as these often vary strongly over their vertical extent. Consequently it is often important to make measurements at multiple locations vertically. This presents a problem for unattended monitoring. One solution is to purchase multiple instruments and position them at intervals throughout the range of interest. This can be very expensive and still yield inadequate vertical resolution. As technology has progressed, so too has the ability to automate the movement and positioning of a single point instrument. An intelligent winch can be used to precisely position sensors and to continuously pass along their real-time data stream. Designed to be a component of a complete vertical profiling system, the winch will provide an interface to its internal computer that will accept commands instructing the system to move the instrument to a particular position and at what speed. Position can be specified by number of winch drum revolutions from a known home position (in very small increments). Additionally, if external position feedback is provided to the system (for example depth in water from an external transducer) position can be specified in the units of the external feedback. The system can be protected from malfunction by monitoring the cable tension via additional optional inputs.

Description

The system consists of a winch drum whose diameter, length, and side flange height are determined by the turning radius and amount of electrical/lifting cable to be spooled. The drum is mounted on a frame via bearings and a hollow axle that can be turned by an electric motor from one side and has a slip ring installed on the other. The motor is sized for the expected load, geared for the desired maximum rotational speed, and has an integral position encoder. The electrical/lifting cable is specified considering the expected mechanical load as well as the number and gauge of conductors that will be required to provide power and communications to the attached instrumentation. A weather-tight box mounted to the frame includes a computer and motor controller, as well as connectors for the cable/slip ring contacts, an Ethernet connector for the internal computer interface and digital I/O ports for the high/low limit/tension inputs and manual up/down controls.

The software interface provides a simple command set to control winch movement. In the absence of optional position feedback, commands would be for up and down by a number of winch revolutions at a specified speed. Given external position feedback additional commands become available to move to a specific position or to move a specific distance. Commands are also available to provide a motor current limit and to stop the system. The system will provide feedback about position, moving speed (or stopped), motor current draw, and fault conditions including over-current, over-temperature, and high and low limit inputs.

Innovation

The intelligent winch was designed from the ground up to be a computer-controlled component of a larger automation system. No other device found in this application area has a simple computer interface for payload positioning. Payload position feedback to the control computer has not been found anywhere else. This device solves the problem of positioning scientific instrumentation vertically to make measurements in profile. Payload position feedback increases accuracy of vertical positioning which can be inaccurate when moving based on drum revolutions due to changing diameter as the cable is wound on the drum one layer after another, or when moving based on cable payout due to currents which may cause the instruments to hang at an angle from vertical. The original design goals were centered on the application of profiling with water quality instrumentation, however future applications include use in other fluids (in tanks or wells, in the atmosphere) to move and position a payload as part of an automated system.

Software Manual

Upon power-up the system first reads the IP address, DNS, gateway, subnet mask and TCP port base values that are stored in EEPROM. Default values are:

Variable	Value
IP Address	192.168.0.150
DNS Address	192.168.0.1
Gateway	192.168.0.1
Subnet Mask	255.255.255.0
Port Base	63520

Default values can be restored by powering on the system while depressing both the external up and down buttons simultaneously. These values can also be changed through the command interface.

Other stored values are then read from EEPROM including:

- Maximum number of revolutions of the winch drum
- PID tuning: P value
- PID tuning: I value
- PID tuning: D value
- PID tuning: PID Scalar
- PID tuning: Velocity feed forward value
- Total motor run time

The above values are *not* restored to defaults when resetting the TCP/IP configuration; however the as-delivered values are indicated on a separate sheet.

Connecting

Interfacing to the system is done via a telnet connection to three different port addresses:

Port Address	Description
Port Base + 0	**Output port:** comma delimited lines of output at 1 Hz
Port Base + 1	**Command port:** accepts system commands and issues replies
Port Base + 2	**Position feedback port:** reads values, one per line, indicating position of payload

Note: Connections are not established until the connecting system sends at least one character (e.g. a newline).

Output data

Upon connecting to the output port, lines of data are returned by the system at 1 Hz. The data are comma separated. The values are as follows (in order):

Item	Units	Description
Position	Revolutions	Position of drum in revolutions from zero. Negative values

		indicate up from zero, positive indicates down.
Velocity	Percent of Max	Negative values indicate moving up, positive indicates down. Can exceed 100 if drum is turning faster than specified maximum speed.
Motor Current	Amps	Motor current draw. Useful for diagnostics, and for choosing a reasonable maximum current limit.
No-Raise	Boolean	Asserted (1) when No-Raise digital input is asserted. Prevents motor from moving in the up direction.
No-Lower	Boolean	Asserted (1) when No-Lower digital input is asserted. Prevents motor from moving in the down direction.
E-Stop	Boolean	Asserted (1) when the emergency stop button is pressed. Prevents motor from moving in either direction.
Over Current	Boolean	Asserted (1) when motor current exceeds maximum current limit threshold. This condition causes the motor controller to down-regulate speed. Speed dropping below 90% or target will also stop the motor.
Over Temp	Boolean	Asserted (1) when the motor controller thermistor reads higher than 90°C, cleared when the temperature drops below 80°C. This condition will also stop the motor.

Commands

The command port receives and processes commands given to the system and issues responses to the commands and other readable information such as reasons that the motor was stopped. Upon connection to the command port the system will respond with a banner such as:

```
Welcome to the intelligent winch command server.
```

The following commands are available, and described below:

Network Commands: (Stored in EEPROM)

```
ip_address xxx.xxx.xxx.xxx
dns xxx.xxx.xxx.xxx
gateway xxx.xxx.xxx.xxx
subnet xxx.xxx.xxx.xxx
port_base xxx
```

Motor Commands:

```
stop
move_to_revolution -xxx.xx at xxx
up_revolutions -xxx.xx at xxx
down_revolutions -xxx.xx at xxx
move_to_position -xxx.xx at xxx
up_distance -xxx.xx at xxx
down_distance -xxx.xx at xxx
set_zero
max_revolutions xxx.xx  (Stored in EEProm)
amps_limit x.xx
motor_cpr (Stored in EEProm)
motor_rpm (Stored in EEProm)
```

PID Tuning Commands:

```
pterm x
iterm x
dterm x
pidscalar x
vff x
store_tuning
```

System Commands:

```
fb_period x.xx
save_state
reboot
motor_time x.xx
reset_motor_time
ver
uptime
clearreports (internal)
```

ip_address xxx.xxx.xxx.xxx
dns xxx.xxx.xxx.xxx
gateway xxx.xxx.xxx.xxx
subnet xxx.xxx.xxx.xxx

These commands set the IP address, DNS, gateway, and subnet mask for the system. Values are stored to EEPROM and used on subsequent re-starts. Issuing the command without an argument will return the current value. These values can be restored to their defaults by holding the up and down buttons in for a few seconds during system start-up. A tutorial for proper TCP/IP settings will not be given here. Default values are listed above. For debugging/testing purposes, if the IP address is set to 0.0.0.0 the system will use DHCP to obtain an address. You will need a way of knowing what address was assigned to the system in order to connect (possibly read it from a router table, etc).

port_base xxx

This command sets the base value for the three TCP ports to be used to communicate with the system. The value is stored to EEPROM and used on subsequent re-starts. Issuing the command without an argument will return the current value. Port values are assigned as described above in the *Connecting* section.

Port values range from 0-65535, however certain values are considered more acceptable than others. Values from 0-1023 are considered “well-known” and should be avoided if possible. Values from 1024-49151 are generally acceptable for use, however many of these values are registered with the Internet Assigned Numbers Authority and may overlap with some known applications. Values from 49152-65535 are dynamic and private ports and should be safe to use.

stop

Stop the motor under any circumstance (even with button pressed).

move_to_revolution -xxx.xx at xxx
up_revolutions -xxx.xx at xxx
down_revolutions -xxx.xx at xxx

Move by monitoring revolutions of the drum. The **move_to_revolution** command will go to a specified drum position relative to the zero point. Negative values indicate a position up from the zero point and positive values indicate a position down from the zero point. "at xxx" indicates the speed to move given as a percentage of the maximum speed. Values of speed range from 2-100 whether or not the motor is capable of that many different speeds. The speed value reported in the output will reflect the actual motor speed. The **up_revolutions** and **down_revolutions** commands are measured relative to the starting position.

move_to_position -xxx.xx at xxx
up_distance -xxx.xx at xxx
down_distance -xxx.xx at xxx

Move by monitoring position feedback on the feedback port. The **move_to_position** command will move until a specified feedback position is met or exceeded. The move will be up if the requested position is less than the current position and vice-versa. "at xxx" indicates the speed to move given as a percentage of the maximum speed. Values of speed range from 2-100 whether or not the motor is capable of that many different speeds. The speed value reported in the output will reflect the actual motor speed. Speeds are limited by the feedback period so that the drum cannot turn more than a quarter turn between feedback position updates. The **up_distance** and **down_distance** commands are measured relative to the starting position. In this mode, if the feedback is not recent, not changing, or changing in the direction opposite to what was expected the move will be stopped.

set_zero

This command sets the current winch position in revolutions to zero. Positions are measured in revolutions relative to the zero position.

max_revolutions xxx.xx

Movements will not be allowed to exceed the max revolutions value in either the positive or negative direction. The max revolutions value is measured relative to the zero position. This can be useful to limit the cable pay-out to approximately the amount of cable available or to limit the payload from exceeding approximately a known depth. The value will be stored in EEPROM and used during subsequent re-starts of the system. Issuing the command without an argument will return the current value. If a movement exceeds the max revolutions value the system will rewind until within limits. Max revolutions cannot be set to less than the current position of the payload.

amps_limit x.xx

This command sets an upper limit to the amount of current that the motor can draw. The value is in amps and must be a positive number up to 14 amps which is the maximum for the motor controller. Issuing the command without an argument will return the current value. Using this command can add another level of safety to the system by setting a limit near the expected current draw for a move at a particular speed. If this limit is exceeded, the movement is stopped. Use of the up or down buttons sets the limit back to the maximum value. Thus it is recommended to set this value before each move.

motor_cpr x

This sets the number of encoder counts per revolution of the drum. The value will be stored in EEPROM and used during subsequent re-starts of the system. Issuing the command without an argument will return the current value. This is a configuration parameter and does not need to be changed unless the motor position encoder is changed.

motor_rpm x.xx

This is the full power RPM at battery voltage (may be higher than the motor spec). This value is used to establish the 100 percent speed of the motor. The value will be stored in EEPROM and used during subsequent re-starts of the system. Issuing the command without an argument will return the current value. This is a configuration parameter that describes properties of the motor; however the value can be set to less than the maximum RPM of the motor to limit the maximum speed, which will consequently limit the maximum current draw.

PID tuning commands

pterm x
iterm x
dterm x
pidscalar x
vff x

Warning: Improper PID tuning can render the system inoperable and possibly even damage hardware! Find PID tuning instructions if necessary in the Solutions^3 Motion Mind 3 datasheet; although, briefly, the pterm and the pidscalar value have the most effect. The as-delivered values are provided on a separate sheet and worked well in testing. PID tuning values will ***not*** be saved to EEPROM until the **store_tuning** command is issued. Issuing one of the above commands without an argument will return the current value.

fb_period x.xx

This command tells the controller the expected period of the position feedback in seconds. During movements that use position feedback, the feedback must be received within 2.5 times the feedback period or the data will be considered too old and the movement will be stopped. The maximum speed during feedback based movements will be limited to the speed at which the feedback period equates to ¼ turn of the winch drum. This will limit the amount of cable payed out in the event that the payload is no longer descending (on an obstruction or on the bottom). The value will be stored in EEPROM and used during subsequent re-starts of the system. Issuing the command without an argument will return the current value.

save_state

This command saves the current state of the system (e.g. position, total run time) to EEPROM so that the system can be completely powered down. When power is restored, the state will be retrieved from

EEPROM and assumed to be correct. The state is also automatically saved if the system remains idle for ten minutes.

reboot
Reboots the system.

motor_time
The motor_time command returns the total accumulated motor run time in days, or sets the value if an argument is given. Monitoring motor run time allows for service to be performed at specific intervals or for pre-emptive replacement of an aging motor. Setting the motor time to a specific value is most useful when replacing hardware on the system (the motor or the computer).

reset_motor_time
Resets the total accumulated motor run time. This is equivalent to 'motor_time 0'.

ver
Returns software version number.

Uptime
The uptime command returns the current accumulated system up time in Days, hours, minutes, seconds and fractions of a second. (e.g. 0 00:00:00.000)

clearreports
Clears the reboot and crash application monitor report. (Used internally for debugging.)

Messages other than responses to commands:
```
Stopped due to NO_RAISE input.
Stopped due to NO_LOWER input.
Stopped due to E-STOP (BRAKE) input.
Stopped due to exceeding motor current limit.
Stopped due to over temperature.
Stopped due to reaching maximum revolution limit.
Stopped due to no recent position feedback.
Stopped due to position feedback value not decreasing during up move.
Stopped due to position feedback value not increasing during down move.
ERROR: Speed not increasing. Possibly lost position encoder data or drum not
turning.
```

Note about commands that write to EEPROM
Some of the EEPROM in use has a rated life of 100,000 writes. Be careful not to issue commands that write to EEPROM too frequently or this may cause irreversible damage to the electronics.

Position Feedback
When position feedback is provided to the system – for example a stream of instrument depth values – additional motor commands become available. Movements of the winch can be specified in the units of position rather than in drum revolutions. This allows for better positioning of the payload for multiple reasons: the effective diameter of the drum changes as the cable is unwound or re-wound; a current can cause the payload to hang at an angle other than exactly vertical. The system will read from the position feedback port one line at a time and expects one number per line. The **move_to_position**, **up_distance**,

and **down_distance** commands will use the position values on the position feedback port to determine when to stop the movement. The position feedback must always be recent. If the age of the feedback exceeds 2.5 times the feedback period the movement will not start or if moving the movement will stop. Position feedback also allows for additional safety checks to be made regarding the movement of the payload. If the movement specifies that the payload should be moving down (or up) and position values are not increasing (or decreasing) within a certain time (equivalent to ¼ revolution of the drum) the movement will be stopped. This helps to prevent problems where the payload is stuck on something or where the cable is winding on the spool in the wrong direction. If the movement is stopped prematurely, a line of information containing the reason will be sent to the command port.

Manual Control
The manual up and down buttons will move the winch over-riding any other move currently in progress. The movement begins slowly and steps up in speed until reaching maximum. The release of either button will stop the move. If the maximum current limit is reached, the system will down-regulate the speed in order to not exceed maximum current. If the speed falls below 50% of the intended value the system will stop the move.

No-Raise & No-Lower Inputs
Two digital inputs are provided that when asserted (connected to positive – power supply voltage) will prevent the system from raising (NO_RAISE) or lowering (NO_LOWER). These inputs will also immediately stop the motor form moving in the indicated direction. These are highly recommended inputs and can be connected to upper and/or lower limit switches and a high and low cable tension monitoring device. Monitoring cable tension and limits can prevent costly problems associated with cable damage such as striking a submerged snag or the bottom and allowing the cable to go slack and come partially off of the drum.

Auto-save state
The system will automatically write the current position and total motor time to EEPROM when the velocity has remained at zero for at least ten minutes. These values will be restored upon re-start. This is very useful under normal operating conditions, but can be surprising during set-up and maintenance. If the payload has moved without this value being reset, the zero point can be reset using the **set_zero** command.

Logging
If the system is started after a micro-SD card has been inserted in the slot on the center card, all traffic to and from the Ethernet ports and the internal (debugging) serial port are logged to a set of rotating log files. Log files are numbered from 00 to 99 and the numbers are incremented at a fixed interval (daily) or with each re-start. If a file was not used the number will not increment, therefore the numbers may not match for the same period of time among the various files. After log 99 the numbers reset to 00 and a one file number gap is maintained to indicate which file is most recent. (E.g. If all 100 files have been created and the current file is numbered 10, then file 11 will have been removed.) If the SD card fills before all files have been created the system will maintain as many logs as possible to leave a specific amount of free space for new log data. Log file data are time stamped with the system uptime.

TCP Keepalive

In order to maintain connectivity to the TCP ports (in the event of a remote system crash or network cable disconnection for example) the command and position feedback ports send TCP keepalive packets on a regular basis. (The output port is already sending characters on a regular basis.) These are empty packets and will have no effect under normal circumstances except to consume a small amount of bandwidth. Several seconds after a network disruption (about 30s), the system will recognize that the ports have become disconnected. If this occurs the connecting system will have to re-connect to the ports including sending an initial character again.

Updating Firmware

New firmware is distributed in the form of a '.hex' file. This file is loaded into the system via USB using the utility XLoader (available at: http://russemotto.com/xloader/ as of this writing) which runs under windows.

- Plug the USB cable into the controller and to a windows computer
- You may need to install driver for USB
- Run XLoader.exe
- Select the .hex file in the first dialog box.
- Under device, select 'Mega (ATMEGA2560)'
- Select correct com port
- Baud rate should be automatically set
- Click upload