UniBot V 0.1A

Creative Robotics Ltd 2014

support@creative-robotics.com

Online documentation and support

For the latest version of this document and for other technical information, schematics and software please visit:

http://www.creative-robotics.com/unibot

Wireless Serial Control

The robot uses a XBee wireless module configured as a transparent serial link. The module is an XBee Pro, Series 1 and more details are available from here:

http://goo.gl/fXJw1B

At the robot end the radio modules communicates with the robot via a 3.3v serial port. At the host computer end the radio module includes a serial to USB converter and will plug into the USB port.

The correct USB serial port drivers must be installed. The USB to serial converter is an FTDI FT231X and the required drivers are available from here:

http://goo.gl/VtwpmH

Power System

The robot runs from a 12V battery and has a hardware safety system that can cut off power to the motors and steering actuators whilst retaining power for the control computer.

The robot is switched on with a large rocker switch on the right hand control panel. The green LED will light when power is on.

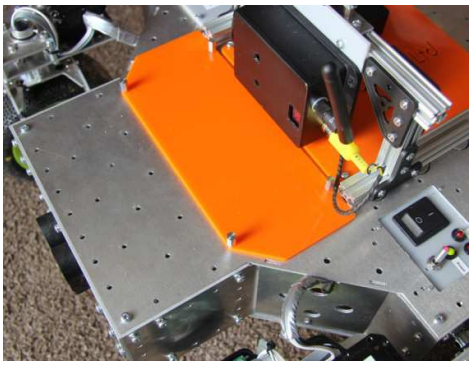
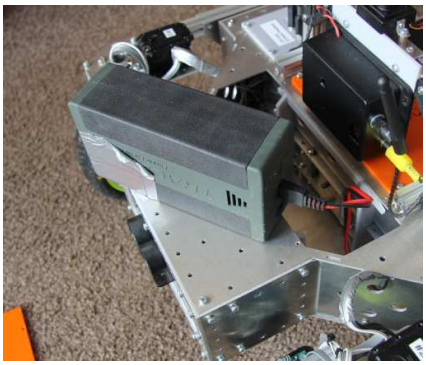
An additional toggle switch with a red LED in the tip controls the safety cut off system. When this is activated a red LED on the control panel will light and the robots motors will receive power. Switching this toggle switch off will disconnect power from the motion system.

CAUTION: When the robots safety cut off is disabled and power is supplied to the motors it will move its steering motors into a home position.

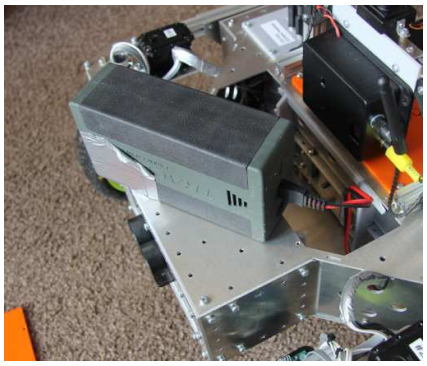
Battery

The robot uses a Tracer Lithium battery supplied by Deben. The battery generates 12V and can supply a maximum of 10A. The battery incorporates its own charging electronics and a bettery level meter.

To remove the battery, unscrew the four retaining screws on the battery cover at the front of the robot.

Lift the battery out of the robot. Charge level can be tested by pressing a button and observing the LED display.




Theory of operation:

The robot has four wheels, each independently driven and steered. UniBot 0.1 is able to operate the independent steering of each wheel through +/- 150 degrees.

In order to perform a turn the robot must turn each wheel such that the axle of each wheel points to the same spot. The robot is then able to drive in an arc about this point.

In order to drive around this point each wheel must be set to the correct speed, determined by the distance of each wheel from that point.

In a robot where the steering has an unrestricted angular range it is possible to produce fully holonomic motion where the robot can move in any direction, drive in an arc around any point, alter its orientation with respect to the environment and transition between any form of motion seamlessly.

Due to the limits of angular steering range in UniBot 0.1 this fully holonomic drive is not possible and there are circumstances where the steering must perform a large rotation when moving between positions that are close together.

An example of this is where the robot is driving around point located at distance X from the robots centre. The value of X can be reduced to zero, placing the point in the centre of the robot. If X is taken from a positive value to a negative value then the wheels must perform a large steering adjustment and the turning point transitions from one side of the robot to the other.

UniBot 0.1A Serial Command Set:

Serial Port Settings:
57600 Baud
1 Stop bit, no parity

All communication is in data packets that begin with the # character and are terminated with a newline character.

A single command character after the packet start character determines the type of data packet. There are two basic packet types, a command packet where data is sent to the robot and a reply received, and a message packet where a text string is sent from the robot. Text strings are indicated by the > command character

A command message starts with # and is followed by one command character. This can then be followed by a series of data values (floating point, single precision) separated by spaces.

Example 1:
To make the robot to drive straight ahead in differential steering mode send the following text string terminated with the line feed character:
#D1.0 1.0

Example 2:
To halt the robot send the following text string terminated with the line feed character:
#X

Example 3:
To request the state data from wheel number 2 send the following text string terminated with the line feed character:
#W2

It is possible to send fewer values than specified by the command and still have the command and data recognised, for example motion commands have the robot speed as their first data value, followed by other values relating to turning for different drive modes. Sending any drive command with just the robot speed value will update the robot speed without changing the other parameters.

Example:
Send: #D1.0 0.0
This will set the speed to 1 and drive the robot straight ahead
Send: #D0.5
This will update the speed to 0.5 but leave the steering unchanged

Returned Data

By default the robot will reply to any motion command with a data packet. A number of different packets can be selected as the default reply and any data packet that has been defined can be requested with a serial command

Examples:
Send: #t4 to select 'data packet 4' as the default reply
After this, whenever a motion command is send you will get data packet 4 as the reply

Send: #!2 To get 'data packet 2'
This will not change the default reply type from 4 to 2 but will return the contents of packet 2

Startup:
The robot boots up and performs a system check. If the check is OK and all the safety systems are off it will initialise the motors and steering, and align the wheels.

The robot boots up into pause state 1 where power is applied to the motors but it will NOT respond to motion commands.

Before the robot can move it must be placed in pause state 0
The robot can be halted at any time by placing it in pause state 1 or by calling the emergency stop function.

Emergency Stop:
If the robot is halted by an emergency stop command it will disconnect power from the wheels and steering. These systems will re-initialise when the robot is placed back into pause state 0

An illuminated toggle switch on the robot body can also trigger an emergency stop – this will disconnect power to the motors and the robot will only begin responding once the switch has been released. This function does not affect the pause state.

IMPORTANT: When released from an emergency stop the robot will re-align its wheels.

Motion Commands and Drive Modes

Drive modes determine how the robot is controlled and different modes can emulate different wheel morphologies. Modes are implicitly selected by sending a command with data. An additional command can limit the speed at which the steering will operate.

The wheels can operate in two modes, one with closed loop PID speed control and the other without.

Set Steering Motor Speed

Command Char: 'g'
ASCII Decimal Value: 103
Parameters:
Data Value: Motor Speed
Data Range: 0.0 to 1.0

Returns:
ACK
Description:
Sets the response speed of the steering motors. A value of 1.0 sets the servo motors to the fastest speed and 0.0 will stop them responding at all.

Set Wheel Control Mode

Command Char: 'M'
ASCII Decimal Value: 77
Parameters:
Data Value: Mode selection
Data Range: ASCII character '0' or '1'

Returns:
ACK or NACK if incorrect data value sent
Description:
This will set each wheels control mode

Sending the character '0' sets the wheels to raw power mode (no speed control)

Sending the character '1' sets the wheels to PID Mode.

Differential Drive

Differential Drive External

Command Char: 'D'
ASCII Decimal Value: 68
Parameters:
Data Value 1: Robot Speed (The forward/reverse speed)
Range: -1.0 to 1.0
Data Value 2: Left/Right Speed Difference
Range: -1.5 to 1.5

Returns:
Four encoder readings (Long ints) and then four sonar readings (short ints) Each are sent in wheel order (0-3) with the sonars numbered according to the wheels they are closest to (Labels on the wheels are 1-4 and map to wheel numbers 0-3).

Description:
This emulates a symmetrical differential drive robot and can turn about a point located outside the body of the robot when between -1.0 and 1.0 (where 0.0 is infinity). Values above 1.0 (or below -1.0) put the turning point inside the robot body but constrained to the left or right hand side. Positive values place this turning point on the right of the robot, negative values on the left.

Differential Drive Internal

Command Char: 'd'
ASCII Decimal Value: 100
Parameters:
Data Value 1: Robot Speed (The rotational speed)
Range: -1.0 to 1.0
Data Value 2: Left/Right Speed Difference
Range: -1.0 to 1.0

Returns:
Four encoder readings (Long ints) and then four sonar readings (short ints) Each are sent in wheel order (0-3) with the sonars numbered according to the wheels they are closest to (Labels on the wheels are 1-4 and map to wheel numbers 0-3).

Description:
This emulates a symmetrical differential drive robot that is turning about a point located inside the body of the robot. A speed difference range of zero will cause the robot to rotate about its central point.

Ackermann Drive

This emulates a 'car style' steering system where wither the front or rear wheels have steering and the other pair are fixed. There are two modes, one that is fixed to the robots own morphology and another where the location of an imaginary axle for the non-steering wheels is located. This can be used to emulate a different sized platform, for example to reproduce the motion of a sensor located at a specified point on a vehicle.

Ackermann Drive Variable

Command Char: 'C'
ASCII Decimal Value: 67
Parameters:
Data Value 1: Robot Speed (The drive speed)
Range: -1.0 to 1.0
Data Value 2: Turn Rate
Range: -1.0 to 1.0

Returns:
Four encoder readings (Long ints) and then four sonar readings (short ints) Each are sent in wheel order (0-3) with the sonars numbered according to the wheels they are closest to (Labels on the wheels are 1-4 and map to wheel numbers 0-3).

Description:
This emulates Ackermann steering ('Car' type steering) where the location of the steering axle is a variable set using another command (You can set the axle to a different location than the physical robot wheel and emulate a different wheel base)

Ackermann Drive (Front Steering)

Command Char: 'A'
ASCII Decimal Value: 65
Parameters:
Data Value 1: Robot Speed (The drive speed)
Range: -1.0 to 1.0
Data Value 2: Turn Rate
Range: -1.0 to 1.0

Returns:
Four encoder readings (Long ints) and then four sonar readings (short ints) Each are sent in wheel order (0-3) with the sonars numbered according to the wheels they are closest to (Labels on the wheels are 1-4 and map to wheel numbers 0-3).

Description:
This emulates Ackermann steering ('Car' type steering) where the location of the steering axle is in-line with the front wheels.

Ackermann Drive (Rear Steering)

Command Char: 'a'
ASCII Decimal Value: 97
Parameters:
Data Value 1: Robot Speed (The drive speed)
Range: -1.0 to 1.0
Data Value 2: Turn Rate
Range: -1.0 to 1.0

Returns:
Four encoder readings (Long ints) and then four sonar readings (short ints) Each are sent in wheel order (0-3) with the sonars numbered according to the wheels they are closest to (Labels on the wheels are 1-4 and map to wheel numbers 0-3).

Description:
This emulates Ackermann steering ('Car' type steering) where the location of the steering axle is in-line with the rear wheels (Like a fork lift truck)

Set Axle Position

Command Char: 'z'
ASCII Decimal Value: 122
Parameters:
Data Value 1: Axle Position
Range: any floating point value

Returns:
Ack or Nack

Description:
Sets the virtual axle position for Ackermann steering mode. Units are in cm so sending 15.2 will set the drive axle to 15.2cm in front of the robot.

Vector Drive

Vector drive is the most versatile drive mode. In this mode a position is specified relative to the centre of the robot and the robot can drive around this point. Polar co-ordinates are used to specify a position by distance and angle. The distance command is a linear value scaled to between -1.0 and 1.0 where 0.0 is infinity with positive and negative values increasing the degree of turn.

Vector Drive

Command Char: 'V'
ASCII Decimal Value: 86
Parameters:
Data Value 1: Robot Speed (The drive speed)
Range: -1.0 to 1.0
Data Value 2: Turn Distance
Range: -1.0 to 1.0
Data Value 2: Turn Angle
Range: -90.0 to 90.0

Returns:
Four encoder readings (Long ints) and then four sonar readings (short ints) Each are sent in wheel order (0-3) with the sonars numbered according to the wheels they are closest to (Labels on the wheels are 1-4 and map to wheel numbers 0-3).

Description:
This defines a point at a distance and angle from the robots body and sets the wheels to drive around this point. The distance value is between infinity (0.0) and a circle within which the robot will sit.

Setting the distance value to 1.0 and angle to 0.0 will cause the robot to turn around a point on the edge of its body to the right hand side.

Setting the distance value to -1.0 and angle to 45 will cause the robot to turn around a point on the edge of its body to the left hand side at an angle of 45 degrees towards the front.

Vector Drive (internal)

Command Char: 'v'
ASCII Decimal Value: 118
Parameters:
Data Value 1: Robot Speed (The rotation speed)
Range: -1.0 to 1.0
Data Value 2: X Distance
Range: -1.0 to 1.0
Data Value 2: Y Distance
Range: -1.0 to 1.0

Returns:

Four encoder readings (Long ints) and then four sonar readings (short ints) Each are sent in wheel order (0-3) with the sonars numbered according to the wheels they are closest to (Labels on the wheels are 1-4 and map to wheel numbers 0-3).

Description:
This defines a point (X and Y) from the centre of the robots body and sets the wheels to drive around this point. The distance value is scaled to the inside dimensions of the robot footprint (the locations of the four steering axis)

Setting X to 1.0 and Y to 0.0 will cause the robot to turn around a point directly in between the front wheels

General Commands

Ping Robot

Command Char: '?'
ASCII Decimal Value: 63
Parameters:
None

Returns:
ACK

Description:
Check that the robot is responding to commands

Set Pause State

Command Char: 'P'
ASCII Decimal Value: 80
Parameters:
Data Value: Wheel Number
Data Range: ASCII Character between '0' and '9' to set the pause state.

Returns:
ACK if a valid pause state char was received or NACK if the value was out of range.

Description:
Pause can set the robot to different states:
'0' Normal Operation
'1' Suspended operation – motors disabled

Emergency Stop

Command Char: 'X'
ASCII Decimal Value: 88
Parameters:
None

Returns:
ACK
Description:
Halts the robot – Disconnects power from the motors and steering servos and sets the pause state to '2'. To restart the robot set the pause state to zero.

Robot sensor data packets

There are a number of pre-defined data packets that the robot can send. The motion commands will also respond with one of these data packets depending on what has been configured.

An additional command and value can be sent to request a status packet from a selected wheel.

Get Robot Data Packet

Command Char: '!'
ASCII Decimal Value: 33
Parameters:
Data Value: Data packet number
Data Range: 0 - 9

Returns:
The specified data packet or NACK if data value was out of bounds
Description:
Requests a data packet from the robot. There are ten possible data packets.

Set Data Packet Type

Command Char: 't'
ASCII Decimal Value: 116
Parameters:
Data Value: Data packet number
Data Range: 0 - 9

Returns:
ACK if OK, NACK if data value was out of bounds
Description:
Sets the data packet type that any motion command will return. When you send any motion command to the robot it will reply with data, the type of data is determined by the data packet type value and can be set here.

NOTE: The default packet type is 1

Data Packet Types

0 ACK (Just replies with an ACK)
1 Encoder and Sonar data (4 encoder values followed by 4 sonar values)
2 All four encoder values
3 All four sonar values
4 All the values captured by the Rangefinder in scanning mode
5 IMU Data (Yaw, Pitch, Roll)
6 Not used
7 Not used
8 Not used
9 Not used

Get Wheel Status

Command Char: 'W'
ASCII Decimal Value: 87
Parameters:
Data Value: Wheel Number
Data Range: ASCII Character between '0' and '3' (Maps to wheels 1-4)

Returns:
Four integer values for the specified wheel: Encoder, motor current, supply voltage, board temperature.

Description:
Request state data for the specified drive wheel.

Active Rangefinder

The active rangefinder is a servo mounted laser rangefinder at the front of the robot. It can swing from left to right and capture range data from any of those positions. When using the rangefinder there will be a delay between setting a target angle and the positioning system achieving that angle so it is desirable to set the angle first, then issue the read data command after approximately 100 milliseconds.

Set Rangefinder

Command Char: 'Q'
ASCII Decimal Value: 81
Parameters:
Data Value: Angle in degrees
Data Range: -60.0 to 60.0

Returns:
ACK

Description:
Sets the rangefinder to the specified angle

Get Range

Command Char: 'q'
ASCII Decimal Value: 113
Parameters:
None

Returns:
A floating point value representing the measured range in mm

Description:
Triggers the Analog to Digital converter to measure the rangefinders output and returns the result.

Start Rangefinder Scanning

Command Char: 'L'
ASCII Decimal Value: 76
Parameters:
None

Returns:
Sets the rangefinder to active scanning mode

Description:
The Rangefinder will scan from side to side, capturing data in an array. The captured data can be retrieved by requesting the correct data type.

Stop Rangefinder Scanning

Command Char: 'l'
ASCII Decimal Value: 76
Parameters:
None

Returns:
Halts active scanning

Description:
Stops the rangefinder if it is in active scan mode.

Inertial Measurement Unit

Get Range

Command Char: 'I'
ASCII Decimal Value: 73
Parameters:
None

Returns:
Three floating point values send direct from the IMU representing the yaw, pitch and roll

Description:
Requests the latest data from the IMU. The IMU processor is sent the request and the data is routed back as the reply. IMPORTANT: If no IMU is present, or if it is not working properly this command will not return anything. If the IMU has been altered to stream data then this data will appear on the robots main COMM port and corrupt other command packets.