CMU - Robotics Institute **Manipulation Lab**

MLab Simple Hand Documentation

Robbie Paolini <rpaolini@cmu.edu> 2013/07/17

1 Introduction

In this document, we discuss the MLab Simple Hand, and how to control and receive data from it.

1.1 Hardware Description

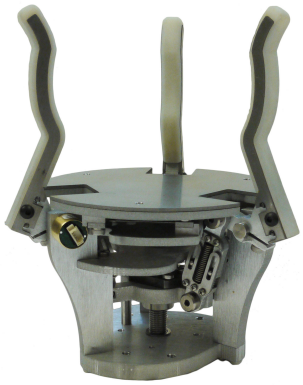

Figure 1: The MLab Simple Hand

The MLab Simple Hand (Figure 1) has 1 motor for actuation connected to 3 compliantly linked fingers. There is an encoder connected to each of the fingers to get angle information. As each finger is connected to the motor with a spring, given the finger position and motor position, the torque exerted by each finger can also be computed. The hand also has a force sensing palm which allows the user to measure the normal force into the palm and 2 torques around the planar directions on top of the palm.

The hand is controlled via a USB connection, and must also be connected to a 6V power supply to power the motor. The hand uses an Arduino as its central processor, and thus code can be modified at any time.

1.2 Angles, Forces, Directions

Here, we explain the meaning of terms used in the rest of the document.

- **Finger Angle:** The angle a finger makes with respect to the hand. As shown in Figure 2, 0 corresponds to the finger being wide open, $\pi/2$ corresponds to the finger being straight up, and π corresponds to the finger being parallel with the palm. Normal values will be around $(\pi/2, \pi)$.
- **Motor Angle:** The motor is controlled by encoder counts, but the *motor angle* is the position of the motor for a nominal finger angle. For example, if we specify a motor angle of $2\pi/3$, this will move the motor so that the finger angles will be $2\pi/3$ in the absence of collision or external forces.
- **Opening and Closing:** As we increase angle, the fingers close. As we increase the motor encoder count, the hand closes as well. It takes a little over 10000 encoder counts to go from fully open to fully closed.

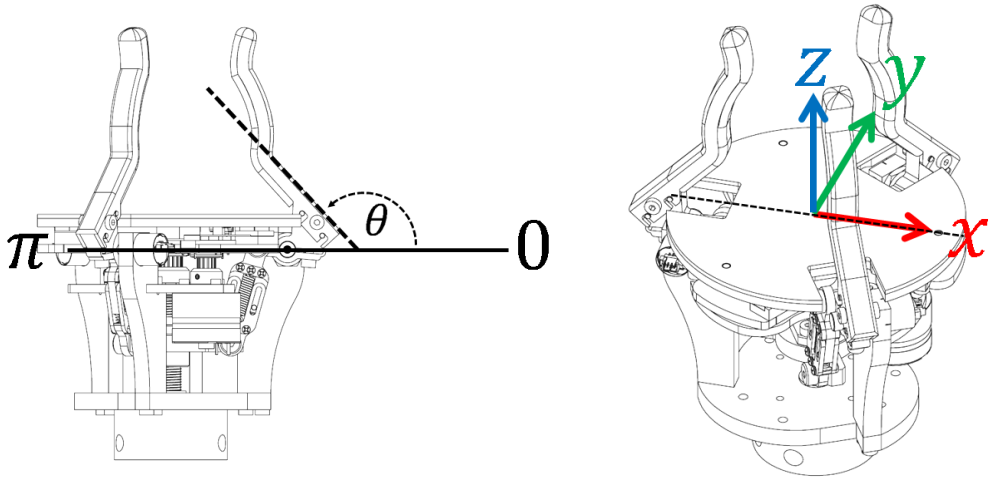

Figure 2: Finger angles and palm frame

- **Finger torques:** As we push or pull the finger from its nominal location, it exerts a force because of the springs compliantly linking it to the motor. This can be expressed as a torque about the finger rotation axis. We will say that the finger torque is positive if the finger is being pressed open, and negative if the finger is being pulled closed.
- **Palm forces and torques:** We can measure planar torques and the normal force using our force sensing palm. As shown in Figure 2, the x-axis corresponds to the vector drawn from the center of the palm out towards the center of the motor (exactly opposite of a finger). The z-axis points outwards, and the y-axis is specified using the right-hand rule. When we report torques, they will be about the x and y axes, and when we report the normal force, it will be in the $-z$ direction.

1.3 State Machine

The hand operates using a very simple state machine, which is shown in Figure 3.

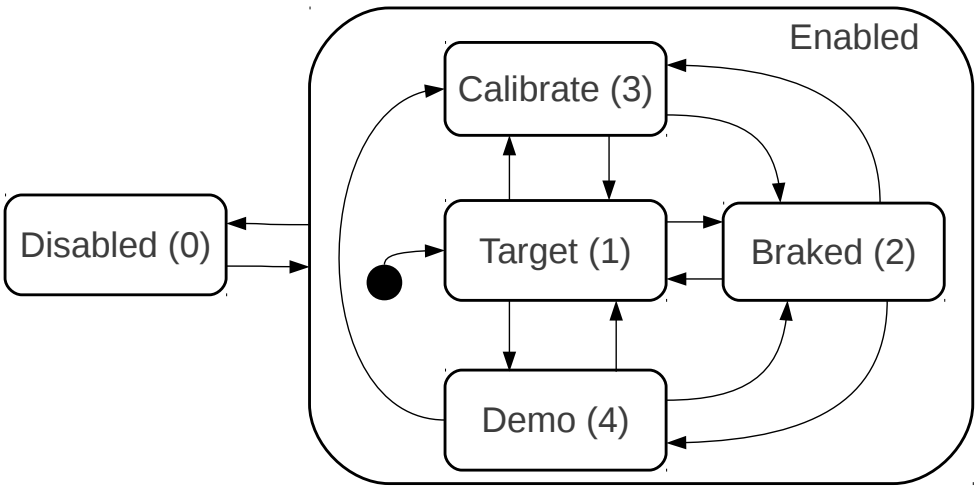

Figure 3: The state machine used by the hand

There are 5 states. Disabled (0), Target (1), Braked (2), Calibrate (3), and Demo (4). The numbers in parentheses next to each state is the value that will be returned when the hand is queried for its current state. The table below explains each of the states in more detail:

State Name	**Description**
Disabled	In this state, the motor controlling the hand is disconnected. It can be moved freely by hand. The hand must be enabled before any motion commands can be executed. The hand can be move to the disabled state from any other state.
Target	This is the main motion state. In this state, the hand is servoed to a target position. If the hand is not calibrated, only encoder target commands can be issued.
Braked	In this state, the hand is braked by "connecting" the leads of the motor together. No power is used and it is harder to turn the motor, but not as strong as when in the Target state.
Calibrate	The hand calibrates itself when in this state. As the motor has an incremental encoder, the encoder is reset by moving the hand to a hard stop. Until the hand is calibrated, finger angles and torques cannot be computed, and no angular motor commands may be issued.
Demo	In this state, we can close and open the hand by pressing on the force sensor. We can only enter this state once the hand has been calibrated.

2 Control Reference

The hand is controlled using a serial interface over the USB connection, with a baud rate of 115200. The hand can accept and reply to commands and queries, stream data, and report errors and information.

2.1 Input

All commands sent to the hand must look like the following:

`command [arg1] [arg2] ... #`

Below we list possible commands to send to the hand with any necessary arguments:

Command	Description
`stop#`	Stops and brakes the motor on the hand.
`disable#`	Disables the motor controller. Before any motion commands are sent, an `enable` command must be sent.
`enable#`	Enables the motor controller and puts the motor in a *floating* state.
`demo#`	Puts hand in demo state.
`calibrate#`	Calibrates the motor on the hand. The hand must be calibrated before motion commands are sent.
`open#`	Opens the hand as wide as it will go.
`open <rad>#`	Opens the hand `<rad>` radians. (rad is a float)
`close#`	Closes the hand as much as possible.
`close <rad>#`	Closes the hand `<rad>` radians. (rad is a float)
`fenc#`	Returns the value of all 3 finger encoders followed by their time stamp.
`fenc <n>#`	Returns the value of finger encoder `<n>` and its time stamp. (n is 0, 1, or 2)
`fang#`	Returns the current angle in radians of all 3 fingers and their time stamps.
`fang <n>#`	Returns the current angle in radians of finger `<n>` and its time stamp. (n is 0, 1, or 2)
`ftorque#`	Returns the current torques on the 3 fingers and their time stamps.
`ftorque <n>#`	Returns the current torque on finger `<n>` and its time stamp. (n is 0, 1 or 2)
`menc#`	Returns the current motor encoder value.
`mang#`	Returns the current motor angle.
`praw#`	Returns the raw readings from the 3 palm sensors and their time stamps.
`praw <n>#`	Returns the raw reading from palm sensor ¡n¿ and its time stamp. (n is 0, 1, or 2)
`pft#`	Returns the force and torques on the palm.
`datadump#`	Returns all information known to the hand at the current time.
`mstatus#`	Returns the motion status of the hand.
`tune#`	Returns the current motor controller gains.
`tune <P> <I> <D>#`	Set the motor controller gains. (P, I, and D are floats)
`speed#`	Returns the current speed of the motor.
`speed <value>#`	Sets the speed of the motor to `<value>`. (Value is an integer between 0 and 100, representing 0-100%).
`current#`	Returns the current limit of the motor.
`current <value>#`	Sets the current limit of the motor to `<value>`. (Value is an integer between 0 and 100, representing 0-100%).
`menc <count>#`	Sets the target and moves the motor to encoder count `<count>`. (count is an integer)
`mang <rad>#`	Sets the target and moves the motor to an angle of `<rad>` radians. (rad is a float)
`stream#`	Returns the current stream register.
`stream <value>#`	Sets up what values to stream. `<value>` is a single character whose 8 bits are a bit array which represent what data to stream.

2.2 Streaming Output

The stream command takes in a single character whose 8 bits are a bit array. Each bit represents a type of data to stream. If it is set to 1, that type of data will be streamed, and if set to 0, it will not. By default, the stream register is initially set to all zeros. We use 8 bits to create our character, with the lowest bit being bit 1, and highest being bit 8:

bit 1	Motor Angle
bit 2	Motor Encoder
bit 3	Finger Angles
bit 4	Finger Encoders
bit 5	Palm Sensors
bit 6	Palm Forces
bit 7	Finger Torques
bit 8	Unused

Based on the current stream register, different data will be sent over the serial port from the hand. All of the data will be sent back with the following form:

`#X<data>&`

where `#` is the start of the mesage, `X` is the data identifier, and `&` is the end of a message.

We list the possible data types and the format of the messages sent back below:

Data Format	Description
`#O<ang>&`	Motor angle (radians). `<ang>` is 4 characters long, representing a number with 3 decimal places. (1.234 → 1234).
`#M<enc>&`	Motor encoder value. `<enc>` is an unsigned integer from 0 to 65536.
`#A<a1><a2><a3>&`	Finger angles (radians). `<a*>` are each 4 characters long, representing a number with 3 decimal places (1.234 → 1234).
`#F<e1><e2><e3>&`	Finger encoder counts. `<e*>` are each 4 characters long, and are integers between 0 and 4095.
`#R<s1><s2><s3>&`	Palm sensor readings. `<s*>` are each 4 characters long, and are integers between 0 and 4095.
`#P<fz><tx><ty>&`	Palm force and torque. Format of `<fz>, <tx>, <ty>` are tbd.
`#T<t1><t2><t3>&`	Finger torques. Format of `<t*>` are tbd.

2.3 Response Output

When a command from the input section is sent to the hand, a response is generated and sent back from the hand. All responses are of the form:

`#QX<message>&`

where `#` is the start of the message, `Q` indicates a response, `X` is the message identifier, and `&` is the end of a message.

Below we list the different responses generated by the hand after different input messages are sent:

Response Format	Description
#QC<success>&	Command confirmation. <success> will be 0 for a failure to execute the requested action, and 1 if the action was successful. stop, disable, enable, calibrate, open, close, tune <P> <I> <D>#, speed <value>#, current <value>#, menc <value>#, mang <value>#, and stream <value># will all generate a command confirmation message.
#QM<enc>&	Motor encoder value. Response generated from a menc# message. <enc> will be an unsigned integer from 0 to 65535 representing the current motor encoder count.
#QO<ang>&	Motor angle. Response generated from a mang# message. <ang> will be a float between 0 at 2π.
#QF<e1>@<t1>,<e2>@<t2>,<e3>@<t3>&	Finger encoders. Response generated from a fenc# message. <e*> will be integers from 0 to 4095, and represent the encoder count of the fingers. <t*> are the timestamps that these values were measured at, and are unsigned long numbers so will range from 0 to 4294967295.
#QN<en>@<tn>&	Finger encoder. Response generated from a fenc <n># message. <en> and <tn> will contain the encoder count and time stamp for finger <n> respectively. Encoder count is an integer between 0 and 4095, and time stamp is an unsigned long from 0 to 4294967295.
#QA<a1>@<t1>,<a2>@<t2>,<a3>@<t3>&	Finger angles. Response generated from a fang# message. <a*> will be floats from 0 to 2π, and represent the angle of the fingers. <t*> are the timestamps that these values were measured at, and are unsigned long numbers so will range from 0 to 4294967295.
#QI<an>@<tn>&	Finger angle. Response generated from a fang <n># message. <an> and <tn> will contain the angle and time stamp for finger <n> respectively. Angle is a float between 0 and 2π, and time stamp is an unsigned long from 0 to 4294967295.
#QT<v1>@<t1>,<v2>@<t2>,<v3>@<t3>&	Finger torques. Response generated from a ftorque# message. <v*> will be floats representing the torque on each of the fingers (units tbd). <t*> are the timestamps that these values were measured at, and are unsigned long numbers so will range from 0 to 4294967295.
#QU<vn>@<tn>&	Finger torque. Response generated from a ftorque <n># message. <vn> and <tn> will contain the torque and time stamp for finger <n> respectively. Torque is a float between with units tbd, and time stamp is an unsigned long from 0 to 4294967295.
#QI<an>@<tn>&	Finger angle. Response generated from a fang <n># message. <an> and <tn> will contain the angle and time stamp for finger <n> respectively. Angle is a float between 0 and 2π, and time stamp is an unsigned long from 0 to 4294967295.
#QR<s1>@<t1>,<s2>@<t2>,<s3>@<t3>&	Palm sensors. Response generated from a praw# message. <s*> will be integers from 0 to 4095, and are each of the palm sensor readings. <t*> are the timestamps that these values were measured at, and are unsigned long numbers so will range from 0 to 4294967295.
#QW<sn>@<tn>&	Palm sensor. Response generated from a praw <n># message. <sn> and <tn> will contain the palm sensor reading and time stamp for sensor <n> respectively. The sensor reading is an integer between 0 and 4095, and time stamp is an unsigned long from 0 to 4294967295.

Response Format	Description
#QP<fz>,<tx>,<ty>&	Palm forces and torques. Response generated from a pft# message. <fz> is the normal force, <tx> is the torque along the x axis of the palm, and <ty> is the torque along the y axis. All values will be floats. units tbd.
#QV<value>&	Current value. Response generated from a speed# or current# message. Returns an integer between 0 and 100 representing the current value of the speed or current limit of the hand.
#QG<P>,<I>,<D>&	Controller Gains. Response generated from a tune# message. <P>, <I>, <D> are the current motor controller gains, and are all floats.
#QM<C>&	Stream register. Response generated from a stream# message. <C> is a single character which represents the 8-bit array of the stream register to see what the hand is currently streaming.
#QS<C>&	Motion Status. Response generated from a mstatus# message. <C> is a single character which represents the 8-bit array of the motion status. See below for an explanation of what each of the bits mean.

The motion status bit-array contains information about the status of the hand, from whether or not it is currently moving, to what state it is in, to whether or not it is calibrated. The meaning of the bits are shown below:

bit 1	1 if hand is moving, 0 otherwise
bits 4-2	Current state we're in. (See state machine explanation.)
bit 5	1 if hand is calibrated, 0 otherwise
bits 8-6	Unused

The only command not covered in our table of responses is the datadump# command. This will send a large amount of text over the serial port, and will be sent as an information message, which is covered in the next section.

2.4 Other Messages

In addition to the messages mentioned above, there are several more that the hand can send over serial:

Message Format	Description
#E<code>&	Error message. <code> is the integer code of the error received.
#I<message>&	Information message. <message> is a string of variable length that does not contain the & character.

Different error codes are listed below:

Error Code	**Description**
00	No Error
01	Unrecognized Command
02	Hand Not Calibrated Error
03	Illegal State Change