

## Chapter 1

# HTTP API

# EiT API

---

## Device status

Resource for storing and fetching the status of a device with a given id.

### Get device status

#### 0.1 GET /status/{device}

##### REQUEST

*raw*

##### RESPONSE

200 (OK)  
Content-Type: application/json

```
{
  "timestamp": "Timestamp in milliseconds when the server received the last status
update",
  "_id": "Database id, not needed for anything",
  "device_id": "The same as the {device}-part of the request",
  "data1": "3.141529",
  "data_2": "2.71828",
  "and so on...": "any data the device has sent to the server",
  ...
}
```

## Set device status

#### 0.2 POST /status/{device}

##### REQUEST

*raw*

Content-Type: application/json

```
{
  "data1": "3.141529",
  "data_2": "2.71828",
```

```
"and so on...": "any data here will be stored by the server",
...
}
```

#### RESPONSE

200 (OK)  
Content-Type: application/json

Will return the same as a GET request to [/command/{device}]

## Manage sensor data for a single sensor

Resource for storing and fetching sensor data for a given sensor for a given device.

### Get sensor data

#### 0.3 GET /data/{device}/{sensor}

##### REQUEST

*raw*

##### RESPONSE

200 (OK)  
Content-Type: application/json

```
{
  "timestamp": "Timestamp in milliseconds when the server received the last status
update",
  "_id": "Database id, not needed for anything",
  "device_id": "The same as the {device}-part of the request",
  "sensor": "The same as the {sensor}-part of the request",
  "any_key": "data specified by the device when updating the sensor data",
  ...
}
```

### Set sensor data

#### 0.4 POST /data/{device}/{sensor}

##### REQUEST

*raw*

Content-Type: application/json

```
{
  "any_key": "data specified by the device when updating the sensor data",
  ...
}
```

#### RESPONSE

200 (OK)  
Content-Type: application/json

Will return the same as a GET request to [/command/{device}]

## Manage sensor data for multiple sensors

Resource for storing and fetching sensor data for all sensors for a given device.

### Get the data from all the device's sensors

#### 0.5 GET /data/{device}

#### REQUEST

*raw*

#### RESPONSE

200 (OK)  
Content-Type: application/json

```
[
  {
    update",
    "timestamp": "Timestamp in milliseconds when the server received the last status
    "_id": "Database id, not needed for anything",
    "device_id": "The same as the {device}-part of the request",
    "sensor": "The id of this sensor",
    "any_key": "data specified by the device when updating the sensor data",
    ...
  },
  {
    update",
    "timestamp": "Timestamp in milliseconds when the server received the last status
    "_id": "Database id, not needed for anything",
    "device_id": "The same as the {device}-part of the request",
    "sensor": "The id of this sensor",
    "any_key": "data specified by the device when updating the sensor data",
    ...
  },
  ...
]
```

## Set the data for several of the device's sensors

### 0.6 POST /data/{device}

#### REQUEST

*raw*

Content-Type: application/json

```
[
  {
    "sensor": "The id of this sensor",
    ...
  },
  {
    "sensor": "The id of this sensor",
    ...
  },
  ...
]
```

#### RESPONSE

200 (OK)

Content-Type: application/json

Will return the same as a GET request to [/command/{device}]

## Manage a device's command queue

Resource for adding commands to a device's command queue and retrieving the command queue.

### Get the device's command queue and flush it

### 0.7 GET /command/{device}

#### REQUEST

*raw*

#### RESPONSE

200 (OK)

Content-Type: application/json

```
[
  {
    "timestamp": "Timestamp in milliseconds when the server received the last status
update",
    "any_key": "Any data can go here",
    ...
  },
  ...
]
```

## Add a command to the device's command queue

### 0.8 POST /command/{device}

#### REQUEST

*raw*

Content-Type: application/json

```
{
  "any_key": "Any data can go here",
  ...
}
```

#### RESPONSE

200 (OK)  
Content-Type: application/json

```
{}
```

# Chapter 2

## Agent code

### 2.1 car.h

```
1 #ifndef CAR_H
2 #define CAR_H
3
4 #include "motor.h"
5 #include <pthread.h>
6
7 #define NO_TURN 0
8 #define LEFT_TURN 1
9 #define RIGHT_TURN 2
10
11 #define TURN_MAGNITUDE 0.5 f
12
13 class Car{
14
15 public:
16     Car(int FR,int FL,int BR,int BL) : frontRightWheel(FR, WHEELMODE), frontLeftWheel(FL, WHEELMODE),
17     backRightWheel(BR, WHEELMODE), backLeftWheel(BL, WHEELMODE){turn = NO_TURN; speed = 0; direction = 0; mode = IDLE_MODE
18     };
19     void setSpeed(int, bool);
20     int getSpeed();
21     void turnCar(int);
22     void setMode(int);
23     int getMode();
24     void startPing();
25 private:
26     int direction;
27     int speed;
28     int turn;
29     int mode;
30     Motor frontRightWheel;
31     Motor frontLeftWheel;
32     Motor backRightWheel;
33     Motor backLeftWheel;
34     pthread_t thread_car;
35     static void * staticEntryPoint(void * c);
36     void ping();
37 };
38 #endif
```

include/car.h

### 2.2 car.cpp

```
1 #include "car.h"
2 #include <stdio.h>
3 #include <unistd.h>
4
5 pthread_mutex_t mutex_car = PTHREAD_MUTEX_INITIALIZER;
```

```

6
7 void Car::setSpeed(int theSpeed, bool dir){
8
9     if(getMode() == FAILSAFE_MODE)
10         return;
11
12     try{
13         switch(turn)
14         {
15             case NO_TURN:
16                 //set all wheels same speed
17                 frontLeftWheel.setSpeed(theSpeed, !dir);
18                 backLeftWheel.setSpeed(theSpeed, !dir);
19                 frontRightWheel.setSpeed(theSpeed, dir);
20                 backRightWheel.setSpeed(theSpeed, dir);
21                 break;
22             case LEFT_TURN:
23                 //set left wheels TURN_MAGNITUDE of right wheels
24                 frontLeftWheel.setSpeed(theSpeed*TURN_MAGNITUDE, !dir);
25                 backLeftWheel.setSpeed(theSpeed*TURN_MAGNITUDE, !dir);
26                 frontRightWheel.setSpeed(theSpeed, dir);
27                 backRightWheel.setSpeed(theSpeed, dir);
28                 break;
29             case RIGHT_TURN:
30                 //set right wheels TURN_MAGNITUDE of left wheels
31                 frontLeftWheel.setSpeed(theSpeed, !dir);
32                 backLeftWheel.setSpeed(theSpeed, !dir);
33                 frontRightWheel.setSpeed(theSpeed*TURN_MAGNITUDE, dir);
34                 backRightWheel.setSpeed(theSpeed*TURN_MAGNITUDE, dir);
35                 break;
36         }
37         speed = theSpeed;
38         direction = dir;
39     }
40     catch(MotorException e) {
41         printf("ID: %d lost\n", e.ID);
42         printError(e.status);
43         setMode(FAILSAFE_MODE);
44         printf("Wheels lost!\n");
45         startPing();
46     }
47 }
48
49 void Car::turnCar(int theTurn){
50
51     if(getMode() == FAILSAFE_MODE)
52         return;
53
54     try{
55         turn = theTurn;
56         if(speed != 0){
57             setSpeed(speed, direction);
58             return;
59         }
60     }
61     if(turn == NO_TURN){
62         setSpeed(0,1);
63         return;
64     }
65     bool dir;
66     if(turn == LEFT_TURN)
67         dir = 1;
68     if(turn == RIGHT_TURN)
69         dir = 0;
70
71     printf("direction %d\n", direction);
72     frontLeftWheel.setSpeed(1023*TURN_MAGNITUDE, dir);
73     backLeftWheel.setSpeed(1023*TURN_MAGNITUDE, dir);
74     frontRightWheel.setSpeed(1023*TURN_MAGNITUDE, dir);
75     backRightWheel.setSpeed(1023*TURN_MAGNITUDE, dir);
76 }
77 catch(MotorException e) {
78     printf("ID: %d lost\n", e.ID);
79     printError(e.status);
80     setMode(FAILSAFE_MODE);
81     printf("Wheels lost!\n");
82     startPing();
83 }
84 }
85
86 void Car::setMode(int theMode){
87     pthread_mutex_lock( &mutex_car );
88     mode = theMode;
89     pthread_mutex_unlock( &mutex_car );
90 }
91
92 int Car::getMode(){
93     pthread_mutex_lock( &mutex_car );
94     int temp = mode;
95     pthread_mutex_unlock( &mutex_car );
96     return temp;
97

```



```

98 }
99
100 void Car::ping(){
101     printf("Ping Car\n");
102     while(1){
103         int count = 0;
104         count += frontLeftWheel.ping();
105         count += backLeftWheel.ping();
106         count += frontRightWheel.ping();
107         count += backRightWheel.ping();
108
109         if(count == 4){
110             printf("All wheels active!\n");
111             setMode(IDLE_MODE);
112             return;
113         }
114     }
115 }
116
117 void Car::startPing(){
118     pthread_create(&thread_car, NULL, Car::staticEntryPoint, this);
119 }
120
121 void * Car::staticEntryPoint(void * c)
122 {
123     ((Car *) c)->ping();
124     return NULL;
125 }

```

src/car.cpp

## 2.3 motor.h

```

1  #ifndef MOTOR_H_
2  #define MOTOR_H_
3
4  #include <dynamixel.h>
5  #include <pthread.h>
6
7  // Control table address
8  #define CW_ANGLE_LIMIT_L 6
9  #define CW_ANGLE_LIMIT_H 7
10 #define CCW_ANGLE_LIMIT_L 8
11 #define CCW_ANGLE_LIMIT_H 9
12 #define MAX_TORQUE_L 14
13 #define MAX_TORQUE_H 15
14 #define HIGH_LIMIT_VOLTAGE 13
15 #define GOAL_POSITION_L 30
16 #define GOAL_POSITION_H 31
17 #define MOVING_SPEED_L 32
18 #define MOVING_SPEED_H 33
19 #define PRESENT_POSITION_L 36
20 #define PRESENT_POSITION_H 37
21 #define PRESENT_SPEED_L 38
22 #define PRESENT_SPEED_H 39
23 #define MOVING 46
24
25 #define WHEELMODE 0
26 #define SERVOMODE 1
27
28 #define CW 1
29 #define CCW 0
30
31 #define IDLE_MODE 0
32 #define FAILSAFE_MODE 1
33
34
35 class MotorException{
36 public:
37     MotorException(int theID, int theStatus) : ID(theID), status(theStatus){};
38     int ID;
39     int status;
40 };
41
42 class Motor{
43 public:
44     Motor(int, int);
45     int getMode();
46     int getPosition();
47     int getSpeed();
48     void setGoalPosition(int);
49     void setSpeed(int, bool);
50     void setMode(int);
51     void setRotateDirection(int);

```

```

52 void printErrorCode(void);
53 void checkStatus();
54 int ping();
55 private:
56 int position;
57 int speed;
58 int mode;
59 int ID;
60 int commStatus;
61 int rotateDirection;
62 };
63
64 void pingAll();
65 void printError(int status);
66
67 #endif

```

include/motor.h

## 2.4 motor.cpp

```

1  #include "motor.h"
2  #include "dynamixel.h"
3  #include "stdio.h"
4  #include "communication.h"
5
6  Motor::Motor(int theID, int theMode){
7      ID = theID;
8      mode = theMode;
9      commStatus = COMM_RXSUCCESS;
10     setMode(mode);
11 }
12
13 int Motor::getMode(){
14     return mode;
15 }
16
17 int Motor::getPosition(){
18
19     int temp = readWord( ID, PRESENT_POSITION_L );
20     commStatus = getResult();
21     if(commStatus != COMM_RXSUCCESS)
22         throw MotorException(ID, commStatus);
23     printErrorCode();
24     position = temp;
25     return position;
26 }
27
28 int Motor::getSpeed(){
29
30     unsigned short temp = readWord( ID, PRESENT_SPEED_L );
31     commStatus = getResult();
32     if(commStatus != COMM_RXSUCCESS)
33         throw MotorException(ID, commStatus);
34     printErrorCode();
35     speed = temp & 1023;
36     return speed;
37 }
38
39 void Motor::setGoalPosition(int thePosition){
40
41     writeWord( ID, GOAL_POSITION_L, thePosition );
42     commStatus = getResult();
43     if(commStatus != COMM_RXSUCCESS)
44         throw MotorException(ID, commStatus);
45     printErrorCode();
46 }
47
48
49 void Motor::setMode(int theMode){
50
51     switch(theMode)
52     {
53     case WHEELMODE:
54         writeWord( ID, CW_ANGLE_LIMIT_L, 0 );
55         writeWord( ID, CCW_ANGLE_LIMIT_L, 0 );
56         break;
57     case SERVOMODE:
58         writeWord( ID, CW_ANGLE_LIMIT_L, 0 );
59         writeWord( ID, CCW_ANGLE_LIMIT_L, 1023 );
60         break;
61     default:
62         printf("unknown mode: %d\n", theMode);
63         return;

```

```

64     }
65     mode = theMode;
66 }
67
68 void Motor::setSpeed(int theSpeed, bool theDirection){
69
70     writeWord( ID, MOVING_SPEED_L, theSpeed | (theDirection << 10) );
71     commStatus = getResult();
72     if(commStatus != COMM_RXSUCCESS)
73         throw MotorException(ID, commStatus);
74     printErrorCode();
75 }
76
77 void Motor::setRotateDirection(int direction){
78
79     switch(direction)
80     {
81     case CW:
82         writeWord(ID, MOVING_SPEED_L, 1024);
83         break;
84     case CCW:
85         writeWord(ID, MOVING_SPEED_L, 0);
86         break;
87     default:
88         printf("invalid input: %d\n", direction);
89         return;
90     }
91     commStatus = getResult();
92     if(commStatus != COMM_RXSUCCESS)
93         throw MotorException(ID, commStatus);
94     printErrorCode();
95
96     rotateDirection = direction;
97 }
98
99 // Print error bit of status packet
100 void Motor::printErrorCode()
101 {
102     if(getRXpacketError(ERRBIT_VOLTAGE) == 1)
103         printf("Input voltage error!\n");
104
105     if(getRXpacketError(ERRBIT_ANGLE) == 1)
106         printf("Angle limit error!\n");
107
108     if(getRXpacketError(ERRBIT_OVERHEAT) == 1)
109         printf("Overheat error!\n");
110
111     if(getRXpacketError(ERRBIT_RANGE) == 1)
112         printf("Out of range error!\n");
113
114     if(getRXpacketError(ERRBIT_CHECKSUM) == 1)
115         printf("Checksum error!\n");
116
117     if(getRXpacketError(ERRBIT_OVERLOAD) == 1)
118         printf("Overload error!\n");
119
120     if(getRXpacketError(ERRBIT_INSTRUCTION) == 1)
121         printf("Instruction code error!\n");
122 }
123
124 void Motor::checkStatus(){
125
126     unsigned char temp;
127     for(int i = 0; i < 50; i++)
128     {
129         if(i == 10 || i == 45)
130             continue;
131         temp = readByte( ID, i );
132         printf("%d:\t%d\t%d\n", ID, i, temp);
133     }
134     printf("\n");
135 }
136
137 int Motor::ping(){
138     pingID(ID);
139     commStatus = getResult();
140     if( commStatus == COMM_RXSUCCESS )
141     {
142         //printf("Motor ID: %d active!\n", ID);
143         return 1;
144     }
145     //printf("Motor ID: %d NOT active!\n", ID);
146     return 0;
147 }
148
149 void pingAll(){
150     for(int i = 0; i < 254; i++){
151         dxl.ping(i);
152         if( dxl.get.result( ) == COMM_RXSUCCESS )
153         {
154             printf("ID: %d active!\n", i);
155         }
156     }
157 }

```

```

156 }
157 }
158
159 void printError(int status){
160     switch(status)
161     {
162     case COMM.TXFAIL:
163
164         printf("COMM.TXFAIL: Failed transmit instruction packet!\n");
165         break;
166
167     case COMM.TXERROR:
168         printf("COMM.TXERROR: Incorrect instruction packet!\n");
169         break;
170
171     case COMM.RXFAIL:
172         printf("COMM.RXFAIL: Failed get status packet from device!\n");
173         break;
174
175     case COMM.RXWAITING:
176         printf("COMM.RXWAITING: Now recieving status packet!\n");
177         break;
178
179     case COMM.RXTIMEOUT:
180         printf("COMM.RXTIMEOUT: There is no status packet!\n");
181         break;
182
183     case COMM.RXCORRUPT:
184         printf("COMM.RXCORRUPT: Incorrect status packet!\n");
185         break;
186
187     default:
188         printf("This is unknown error code!\n");
189         break;
190     }
191 }

```

src/motor.cpp

## 2.5 manipulator.h

```

1 #ifndef MANIPULATOR_H_
2 #define MANIPULATOR_H_
3
4 #include "motor.h"
5 #include <pthread.h>
6
7 #define PI 3.14159265
8
9 #define XSTART 0
10 #define YSTART 155
11 #define ZSTART 77
12
13 class Manipulator{
14 public:
15     Manipulator(int IDOne ,int IDTwo,int IDThree, int IDGrip_left, int IDGrip_right ) :
16         one(IDOne, SERVOMODE), two(IDTwo, SERVOMODE), three(IDThree, SERVOMODE),
17         grip_left(IDGrip_left, SERVOMODE), grip_right(IDGrip_right, SERVOMODE) {theta1 = 0; theta2 = 0; theta3 = 0; mode =
18         IDLE.MODE;};
19     void goToPosition(int, int, int);
20     void setAngles(float, float, float);
21     void setGripper(bool);
22     void drawLine(int, int, int, int, int);
23     void drawCircle(int, int, int, int, float, float);
24     void setMode(int);
25     int getMode();
26     void startPing();
27 private:
28     float theta1;
29     float theta2;
30     float theta3;
31     int mode;
32     Motor one;
33     Motor two;
34     Motor three;
35     Motor grip_left;
36     Motor grip_right;
37     pthread_t thread;
38     static void * staticEntryPoint(void * c);
39     void ping();
40 };
41 #endif

```

---

include/manipulator.h

## 2.6 manipulator.cpp

```

1 #include <stdio.h>
2 #include <unistd.h>
3 #include <math.h>
4 #include "manipulator.h"
5
6 using namespace std;
7
8 #define D2 77 //length of first arm in mm
9 #define D3 155 //length of second arm in mm
10
11 #define ANGLE_TO_VALUE (float)511*6/(5*PI)
12
13 #define GRIPPER_LEFT_ZERO 511-140
14 #define GRIPPER_RIGHT_ZERO 511+140
15 #define MAX_COUNT 5
16
17 pthread_mutex_t mutex_man = PTHREAD_MUTEX_INITIALIZER;
18
19 void Manipulator::goToPosition(int x, int y, int z){
20
21     //return error if beyond max
22     // if ((x*x+y*y+z*z) > (D2+D3)*(D2+D3))
23     // {
24     //     printf("invalid position!\n");
25     //     return;
26     // }
27
28     if(getMode() == FAILSAFE_MODE)
29         return;
30
31     float s3, c3, l;
32
33     l = sqrt(x*x+y*y);
34     c3 = (z*z + l*l - D2*D2 - D3*D3)/(2*D2*D3);
35     s3 = sqrt(1-c3*c3);
36
37     theta3 = atan2(s3, c3);
38     theta2 = PI/2 - atan2(D3*s3, D2+D3*c3)-atan2(z, l);
39     theta1 = atan2(x, y);
40
41     setAngles(theta1, theta2, theta3);
42 }
43
44 void Manipulator::setAngles(float t1, float t2, float t3){
45
46     if(getMode() == FAILSAFE_MODE)
47         return;
48
49     try{
50         int dummy;
51
52         if(t1 != t1)
53             printf("nan theta 1\n");
54         else if(t1 > 5*PI/6){
55             one.setGoalPosition(1023);
56             printf("Theta 1 too high\n");
57         }
58         else if(t1 < -5*PI/6){
59             one.setGoalPosition(0);
60             printf("Theta 1 too low\n");
61         }
62         else{
63             dummy = (float)(t1*ANGLE_TO_VALUE+511);
64             one.setGoalPosition(dummy);
65             //printf("one: %d\n", dummy);
66         }
67
68         if(t2 != t2)
69             printf("nan theta 2\n");
70         else if(t2 > 5*PI/6){
71             two.setGoalPosition(1023);
72             printf("Theta 2 too high\n");
73         }
74         else if(t2 < 0){
75             two.setGoalPosition(511);
76             printf("Theta 2 too low\n");
77         }
78         else{

```

```

79     dummy = (float)(t2*ANGLE.TO.VALUE+511);
80     two.setGoalPosition(dummy);
81     //printf("two: %d\n",dummy);
82 }
83
84 if(t3 != t3)
85     printf("nan theta 3\n");
86 else if(t3 > 0.78*PI){
87     three.setGoalPosition(989);
88     printf("Theta 3 too high\n");
89 }
90 else if(t3 < -0.5*PI){
91     three.setGoalPosition(51);
92     printf("Theta 3 too low\n");
93 }
94 else{
95     dummy = (float)(t3*ANGLE.TO.VALUE+511);
96     three.setGoalPosition(dummy);
97     //printf("three: %d\n",dummy);
98 }
99 }
100 catch(MotorException e) {
101     printf("ID: %d lost\n",e.ID);
102     printError(e.status);
103     setMode(FAILSAFE.MODE);
104     printf("Manipulator lost!\n");
105     startPing();
106 }
107 }
108
109 void Manipulator::setGripper(bool on){
110
111     if(getMode() == FAILSAFE.MODE)
112         return;
113
114     try{
115         if(!on){
116             grip_left.setGoalPosition(511-50);
117             grip_right.setGoalPosition(511+50);
118             return;
119         }
120
121         int positionL, positionR, lastPositionL, lastPositionR;
122         int counter = 0;
123         //put servo set point to zero degrees
124         grip_left.setGoalPosition(GRIPPER_LEFT_ZERO);
125         grip_right.setGoalPosition(GRIPPER_RIGHT_ZERO);
126         lastPositionR = grip_right.getPosition();
127         lastPositionL = grip_left.getPosition();
128         while(1){
129             positionL = grip_left.getPosition();
130             positionR = grip_right.getPosition();
131             printf("left: %d\tright: %d\n",positionL,positionR);
132
133             if(lastPositionL == positionL || lastPositionR == positionR)
134                 counter++;
135             else
136                 counter = 0;
137             if(counter == MAX.COUNT)
138                 return;
139             lastPositionL = positionL;
140             lastPositionR = positionR;
141             usleep(10000);
142         }
143     }
144     catch(MotorException e) {
145         printf("ID: %d lost\n",e.ID);
146         printError(e.status);
147         setMode(FAILSAFE.MODE);
148         printf("Manipulator lost!\n");
149         startPing();
150     }
151 }
152
153 void Manipulator::drawLine(int xstart, int ystart, int xend, int yend, int z){
154
155     if(getMode() == FAILSAFE.MODE)
156         return;
157
158     try{
159         goToPosition(xstart, ystart, z+50);
160         sleep(1);
161         goToPosition(xstart, ystart, z);
162         usleep(100000);
163         int x = xend-xstart;
164         int y = yend-ystart;
165         int length = sqrt(x*x+y*y);
166         x /= length; //normalize
167         y /= length; //normalize
168         for(int i = 0; i<length; i++){
169             printf("x: %d\ty: %d\n",xstart+i*x, ystart+i*y);
170             goToPosition(xstart+i*x, ystart+i*y, z);

```

```

171     usleep(10000);
172 }
173 }
174 catch (MotorException e) {
175     printf("ID: %d lost\n", e.ID);
176     printError(e.status);
177     setMode(FAILSAFE.MODE);
178     printf("Manipulator lost!\n");
179     startPing();
180 }
181 }
182
183 void Manipulator::drawCircle(int xcenter, int ycenter, int z, int radius, float startAngle, float endAngle){
184
185     if(getMode() == FAILSAFE.MODE)
186         return;
187
188     try{
189         float t = startAngle;
190         float stepSize = 0.01;
191         while(t <= endAngle){
192             goToPosition(radius*sin(t) + xcenter, radius*cos(t) + ycenter, z);
193             t += stepSize;
194             usleep(10000);
195         }
196     }
197     catch (MotorException e) {
198         printf("ID: %d lost\n", e.ID);
199         printError(e.status);
200         setMode(FAILSAFE.MODE);
201         printf("Manipulator lost!\n");
202         startPing();
203     }
204 }
205
206 void Manipulator::setMode(int theMode){
207     pthread_mutex_lock(&mutex_man);
208     mode = theMode;
209     pthread_mutex_unlock(&mutex_man);
210 }
211
212 int Manipulator::getMode(){
213     pthread_mutex_lock(&mutex_man);
214     int temp = mode;
215     pthread_mutex_unlock(&mutex_man);
216     return temp;
217 }
218
219 void Manipulator::ping(){
220     printf("Ping Manipulators\n");
221     while(1){
222         int count = 0;
223         count += one.ping();
224         count += two.ping();
225         count += three.ping();
226         count += grip_left.ping();
227         count += grip_right.ping();
228
229         if(count == 5){
230             printf("All manipulator motors active!\n");
231             setMode(IDLE.MODE);
232             //printf("Returning to start position\n");
233             //goToPosition(XSTART,YSTART,ZSTART);
234             //setGripper(0);
235             return;
236         }
237     }
238 }
239
240 void Manipulator::startPing(){
241
242     pthread_create(&thread, NULL, Manipulator::staticEntryPoint, this);
243 }
244
245 void * Manipulator::staticEntryPoint(void * c)
246 {
247     ((Manipulator *) c)->ping();
248     return NULL;
249 }

```

src/manipulator.cpp

## 2.7 sensor.h

```

1  #ifndef SENSOR_H_
2  #define SENSOR_H_
3
4  #include <dynamixel.h>
5
6  //control table address
7  #define IR_LEFT_FIRE_DATA 26
8  #define IR_CENTER_FIRE_DATA 27
9  #define IR_RIGHT_FIRE_DATA 28
10 #define LIGHT_LEFT_DATA 29
11 #define LIGHT_CENTER_DATA 30
12 #define LIGHT_RIGHT_DATA 31
13 #define IR_OBSTACLE_DETECTED 32
14 #define LIGHT_DETECTED 33
15 #define SOUND_DATA 35
16 #define BUZZER_DATA_NOTE 40
17 #define BUZZER_DATA_TIME 41
18
19 #define LEFT 0
20 #define CENTER 1
21 #define RIGHT 2
22
23 /*melody:
24 0: Rising
25 1: Falling
26 2: Fight
27 4: Fail
28 5: sad
29 6: bip bip
30 7: sad 2
31 10: whistle rise
32 11: bip bop
33 15: bip bip 2
34 16: phone
35 21: whistle
36 24: rtrtrtrrt
37 */
38
39 class Sensor{
40 public:
41     Sensor(int);
42     int getIR(int);
43     int getLight(int); //only infrared light
44     void playMelody(int); //input range 0-26
45     void playMelody(unsigned char*, int); //play from arrays in songs.h
46     void ping();
47     void setMode(int);
48     int getMode();
49 private:
50     int ID;
51     int commStatus;
52     int mode;
53 };
54
55 #endif

```

include/sensor.h

## 2.8 sensor.cpp

```

1  #include "motor.h"
2  #include "sensor.h"
3  #include "stdio.h"
4  #include <unistd.h>
5  #include "communication.h"
6
7  Sensor::Sensor(int theID){
8      ID = theID;
9      commStatus = COMM_RXSUCCESS;
10     mode = IDLE_MODE;
11 }
12
13 int Sensor::getLight(int pos){
14
15     int data = readByte( ID, LIGHT_LEFT_DATA + pos );
16     commStatus = getResult();
17     if(commStatus != COMM_RXSUCCESS)
18     {
19         mode = FAILSAFE_MODE;
20         printf("sensor lost\n");
21     }
22     return data;
23 }
24

```



```

25 int Sensor::getIR(int pos){
26
27     int data = readByte( ID, IR_LEFT_FIRE_DATA + pos );
28     commStatus = getResult();
29     if(commStatus != COMM_RXSUCCESS)
30     {
31         mode = FAILSAFE_MODE;
32         printf("sensor lost\n");
33     }
34
35     return data;
36 }
37
38 void Sensor::playMelody(int song){
39
40     if(song < 0 || song > 26){
41         printf("invalid input\n");
42         return;
43     }
44     writeByte(ID, BUZZER_DATA_TIME, 255);
45     commStatus = getResult();
46     if(commStatus != COMM_RXSUCCESS)
47     {
48         mode = FAILSAFE_MODE;
49         printf("sensor lost\n");
50     }
51     writeByte(ID, BUZZER_DATA_NOTE, song);
52     commStatus = getResult();
53     if(commStatus != COMM_RXSUCCESS)
54     {
55         mode = FAILSAFE_MODE;
56         printf("sensor lost\n");
57     }
58 }
59
60 void Sensor::playMelody(unsigned char* song, int length){
61
62
63
64     for(int i = 0; i<length; i+=2)
65     {
66
67         if(song[i+1] != 100)
68         {
69             writeByte(ID, BUZZER_DATA_TIME, 254);
70             writeByte(ID, BUZZER_DATA_NOTE, song[i+1]);
71             usleep(40000*song[i]);
72         }
73         else
74         {
75             writeByte(ID, BUZZER_DATA_TIME, 0);
76             usleep(40000*song[i]);
77         }
78     }
79
80
81     writeByte(ID, BUZZER_DATA_TIME, 0);
82 }
83
84
85 void Sensor::ping(){
86     pingID(ID);
87     commStatus = getResult();
88     if( commStatus == COMM_RXSUCCESS )
89     {
90         printf("Sensor ID: %d active!\n",ID);
91         setMode(IDLE_MODE);
92     }
93     else{
94         setMode(FAILSAFE_MODE);
95     }
96 }
97
98 void Sensor::setMode(int theMode){
99     mode = theMode;
100 }
101
102 int Sensor::getMode(){
103     return mode;
104 }
105

```

src/sensor.cpp

## 2.9 interface.h

```

1 #ifndef INTERFACE_H_
2 #define INTERFACE_H_
3
4 #include "manipulator.h"
5 #include "car.h"
6
7 void windowInit();
8 void checkEvent(Manipulator *, Car *);
9
10 #endif

```

include/interface.h

## 2.10 interface.cpp

```

1 #include <X11/Xlib.h>
2 #include <X11/Xutil.h>
3 #include <stdio.h>
4 #include <stdlib.h>
5 #include "interface.h"
6 #include "manipulator.h"
7
8 #define KEYMASK ButtonPressMask | KeyPressMask | KeyReleaseMask | ButtonReleaseMask | PointerMotionMask
9
10 #define FORWARD 25
11 #define BACKWARD 39
12 #define LEFT 38
13 #define RIGHT 40
14
15 #define LEFT_MOUSE_BUTTON 1
16 #define RIGHT_MOUSE_BUTTON 3
17 #define MOUSE_WHEEL 2
18 #define MOUSE_WHEEL_FORWARD 4
19 #define MOUSE_WHEEL_BACKWARD 5
20
21 Display *display;
22 Window window;
23 XEvent event;
24 bool button = 0;
25 bool buttonR = 0;
26 int xpos = XSTART;
27 int ypos = YSTART;
28 int zpos = ZSTART;
29 int xzero = 0;
30 int yzero = 0;
31
32 void windowInit()
33 {
34     int s;
35     /* open connection with the server */
36     display = XOpenDisplay(NULL);
37     if (display == NULL)
38     {
39         fprintf(stderr, "Cannot open display\n");
40         exit(1);
41     }
42
43     s = DefaultScreen(display);
44
45     /* create window */
46     window = XCreateSimpleWindow(display, RootWindow(display, s), 10, 10, 500, 500, 1,
47                                 BlackPixel(display, s), WhitePixel(display, s));
48
49     /* select kind of events we are interested in */
50     XSelectInput(display, window, KEYMASK);
51
52     /* map (show) the window */
53     XMapWindow(display, window);
54
55     //do not detect autorepeating events from keyboard
56     XAutoRepeatOff(display);
57     printf("Display open\n");
58 }
59 void checkEvent(Manipulator *man, Car *car){
60     XNextEvent(display, &event);
61     switch(event.type){
62     case MotionNotify:
63         if(button){
64             xpos -= event.xmotion.x - xzero;
65             ypos -= event.xmotion.y - yzero;
66             xzero = event.xmotion.x;
67             yzero = event.xmotion.y;
68             //printf("xpos: %d\t ypos: %d\n", xpos, ypos);

```

```

69     man->goToPosition(xpos, ypos, zpos);
70 }
71 break;
72 case ButtonPress:
73     if(event.xkey.keycode == LEFT_MOUSE_BUTTON)
74     {
75         button = 1;
76         xzero = event.xbutton.x;
77         yzero = event.xbutton.y;
78     }
79     if(event.xkey.keycode == RIGHT_MOUSE_BUTTON)
80     {
81         buttonR ^= 1;
82         man->setGripper(buttonR);
83     }
84     if(event.xkey.keycode == MOUSE_WHEEL_FORWARD)
85     {
86         zpos -= 10;
87         man->goToPosition(xpos, ypos, zpos);
88     }
89     if(event.xkey.keycode == MOUSE_WHEEL_BACKWARD)
90     {
91         zpos += 10;
92         man->goToPosition(xpos, ypos, zpos);
93     }
94
95     printf("KeyPress: %d\n", event.xkey.keycode);
96     break;
97 case ButtonRelease:
98     if(event.xkey.keycode == LEFT_MOUSE_BUTTON)
99         button = 0;
100     break;
101 case KeyPress:
102     //printf("KeyPress: %d\n", e.xkey.keycode);
103     switch(event.xkey.keycode){
104         case FORWARD:
105             printf("forward\n");
106             car->setSpeed(1023, 1);
107             break;
108         case BACKWARD:
109             car->setSpeed(1023, 0);
110             printf("backward\n");
111             break;
112         case RIGHT:
113             car->turnCar(RIGHT_TURN);
114             printf("right\n");
115             break;
116         case LEFT:
117             car->turnCar(LEFT_TURN);
118             printf("left\n");
119             break;
120         default:
121             printf("unknown:%d\n", event.xkey.keycode);
122     }
123     break;
124 case KeyRelease:
125     //printf("KeyRelease: %d\n", e.xkey.keycode);
126     switch(event.xkey.keycode){
127         case FORWARD:
128             car->setSpeed(0, 1);
129             printf("forward released\n");
130             break;
131         case BACKWARD:
132             car->setSpeed(0, 1);
133             printf("backward released\n");
134             break;
135         case RIGHT:
136             car->turnCar(NO_TURN);
137             printf("right released\n");
138             break;
139         case LEFT:
140             car->turnCar(NO_TURN);
141             printf("left released\n");
142             break;
143         default:
144             printf("unknown:%d\n", event.xkey.keycode);
145     }
146     break;
147 }
148 }

```

src/interface.cpp

## 2.11 dynamixel.h

```

1  #ifndef _DYNAMIXELHEADER
2  #define _DYNAMIXELHEADER
3
4  #ifdef __cplusplus
5  extern "C" {
6  #endif
7
8
9  /////////////// device control methods ///////////////////
10 int dxl_initialize(int deviceIndex, int baudnum);
11 void dxl_terminate();
12
13
14 /////////////// set/get packet methods ///////////////////
15 #define MAXNUM_TXPARAM (150)
16 #define MAXNUM_RXPARAM (60)
17
18 void dxl_set_txpacket_id(int id);
19 #define BROADCAST_ID (254)
20
21 void dxl_set_txpacket_instruction(int instruction);
22 #define INST_PING (1)
23 #define INST_READ (2)
24 #define INST_WRITE (3)
25 #define INST_REG_WRITE (4)
26 #define INST_ACTION (5)
27 #define INST_RESET (6)
28 #define INST_SYNC_WRITE (131)
29
30 void dxl_set_txpacket_parameter(int index, int value);
31 void dxl_set_txpacket_length(int length);
32
33 int dxl_get_rxpacket_error(int errbit);
34 #define ERBBIT_VOLTAGE (1)
35 #define ERBBIT_ANGLE (2)
36 #define ERBBIT_OVERHEAT (4)
37 #define ERBBIT_RANGE (8)
38 #define ERBBIT_CHECKSUM (16)
39 #define ERBBIT_OVERLOAD (32)
40 #define ERBBIT_INSTRUCTION (64)
41
42 int dxl_get_rxpacket_length(void);
43 int dxl_get_rxpacket_parameter(int index);
44
45
46 // utility for value
47 int dxl_makeword(int lowbyte, int highbyte);
48 int dxl_get_lowbyte(int word);
49 int dxl_get_highbyte(int word);
50
51
52 /////////////// packet communication methods ///////////////////
53 void dxl_tx_packet(void);
54 void dxl_rx_packet(void);
55 void dxl_txrx_packet(void);
56
57 int dxl_get_result(void);
58 #define COMMLTXSUCCESS (0)
59 #define COMMLRXSUCCESS (1)
60 #define COMMLTXFAIL (2)
61 #define COMMLRXFAIL (3)
62 #define COMMLTXERROR (4)
63 #define COMMLRXWAITING (5)
64 #define COMMLRXTIMEOUT (6)
65 #define COMMLRXCORRUPT (7)
66
67
68 /////////////// high communication methods ///////////////////
69 void dxl_ping(int id);
70 int dxl_read_byte(int id, int address);
71 void dxl_write_byte(int id, int address, int value);
72 int dxl_read_word(int id, int address);
73 void dxl_write_word(int id, int address, int value);
74
75
76 #ifdef __cplusplus
77 }
78 #endif
79
80 #endif

```

include/dynamixel.h

## 2.12 dynamixel.c

```

1 #include "dxl_hal.h"
2 #include "dynamixel.h"
3
4 #define ID (2)
5 #define LENGTH (3)
6 #define INSTRUCTION (4)
7 #define ERBBIT (4)
8 #define PARAMETER (5)
9 #define DEFAULT_BAUDNUMBER (1)
10
11 unsigned char gbInstructionPacket[MAXNUM_TXPARAM+10] = {0};
12 unsigned char gbStatusPacket[MAXNUM_RXPARAM+10] = {0};
13 unsigned char gbRxPacketLength = 0;
14 unsigned char gbRxGetLength = 0;
15 int gbCommStatus = COMM_RXSUCCESS;
16 int giBusUsing = 0;
17
18
19 int dxl_initialize(int deviceIndex, int baudnum)
20 {
21     float baudrate;
22     baudrate = 2000000.0f / (float)(baudnum + 1);
23
24     if( dxl_hal_open(deviceIndex, baudrate) == 0 )
25         return 0;
26
27     gbCommStatus = COMM_RXSUCCESS;
28     giBusUsing = 0;
29     return 1;
30 }
31
32 void dxl_terminate(void)
33 {
34     dxl_hal_close();
35 }
36
37 void dxl_tx_packet(void)
38 {
39     unsigned char i;
40     unsigned char TxNumByte, RealTxNumByte;
41     unsigned char checksum = 0;
42
43     if( giBusUsing == 1 )
44         return;
45
46     giBusUsing = 1;
47
48     if( gbInstructionPacket[LENGTH] > (MAXNUM_TXPARAM+2) )
49     {
50         gbCommStatus = COMM_TXERROR;
51         giBusUsing = 0;
52         return;
53     }
54
55     if( gbInstructionPacket[INSTRUCTION] != INST_PING
56         && gbInstructionPacket[INSTRUCTION] != INST_READ
57         && gbInstructionPacket[INSTRUCTION] != INST_WRITE
58         && gbInstructionPacket[INSTRUCTION] != INST_REG_WRITE
59         && gbInstructionPacket[INSTRUCTION] != INST_ACTION
60         && gbInstructionPacket[INSTRUCTION] != INST_RESET
61         && gbInstructionPacket[INSTRUCTION] != INST_SYNC_WRITE )
62     {
63         gbCommStatus = COMM_TXERROR;
64         giBusUsing = 0;
65         return;
66     }
67
68     gbInstructionPacket[0] = 0xff;
69     gbInstructionPacket[1] = 0xff;
70     for( i=0; i<(gbInstructionPacket[LENGTH]+1); i++ )
71         checksum += gbInstructionPacket[i+2];
72     gbInstructionPacket[gbInstructionPacket[LENGTH]+3] = ~checksum;
73
74     if( gbCommStatus == COMM_RXTIMEOUT || gbCommStatus == COMM_RXCORRUPT )
75         dxl_hal_clear();
76
77     TxNumByte = gbInstructionPacket[LENGTH] + 4;
78     RealTxNumByte = dxl_hal_tx( (unsigned char*)gbInstructionPacket, TxNumByte );
79
80     if( TxNumByte != RealTxNumByte )
81     {
82         gbCommStatus = COMM_TXFAIL;
83         giBusUsing = 0;
84         return;
85     }
86
87     if( gbInstructionPacket[INSTRUCTION] == INST_READ )
88         dxl_hal_set_timeout( gbInstructionPacket[PARAMETER+1] + 6 );
89     else
90         dxl_hal_set_timeout( 6 );
91

```

```

92  gbCommStatus = COMM.TXSUCCESS;
93  }
94
95  void dxl_rx_packet(void)
96  {
97      unsigned char i, j, nRead;
98      unsigned char checksum = 0;
99
100     if( giBusUsing == 0 )
101         return;
102
103     if( gbInstructionPacket[ID] == BROADCAST_ID )
104     {
105         gbCommStatus = COMM.RXSUCCESS;
106         giBusUsing = 0;
107         return;
108     }
109
110     if( gbCommStatus == COMM.TXSUCCESS )
111     {
112         gbRxGetLength = 0;
113         gbRxPacketLength = 6;
114     }
115
116     nRead = dxl_hal_rx( (unsigned char*)&gbStatusPacket[gbRxGetLength], gbRxPacketLength - gbRxGetLength );
117     gbRxGetLength += nRead;
118     if( gbRxGetLength < gbRxPacketLength )
119     {
120         if( dxl_hal_timeout() == 1 )
121         {
122             if( gbRxGetLength == 0 )
123                 gbCommStatus = COMM.RXTIMEOUT;
124             else
125                 gbCommStatus = COMM.RXCORRUPT;
126             giBusUsing = 0;
127             return;
128         }
129     }
130
131     // Find packet header
132     for( i=0; i<(gbRxGetLength-1); i++ )
133     {
134         if( gbStatusPacket[i] == 0xff && gbStatusPacket[i+1] == 0xff )
135         {
136             break;
137         }
138         else if( i == gbRxGetLength-2 && gbStatusPacket[gbRxGetLength-1] == 0xff )
139         {
140             break;
141         }
142     }
143     if( i > 0 )
144     {
145         for( j=0; j<(gbRxGetLength-i); j++ )
146             gbStatusPacket[j] = gbStatusPacket[j + i];
147
148         gbRxGetLength -= i;
149     }
150
151     if( gbRxGetLength < gbRxPacketLength )
152     {
153         gbCommStatus = COMM.RXWAITING;
154         return;
155     }
156
157     // Check id pairing
158     if( gbInstructionPacket[ID] != gbStatusPacket[ID] )
159     {
160         gbCommStatus = COMM.RXCORRUPT;
161         giBusUsing = 0;
162         return;
163     }
164
165     gbRxPacketLength = gbStatusPacket[LENGTH] + 4;
166     if( gbRxGetLength < gbRxPacketLength )
167     {
168         nRead = dxl_hal_rx( (unsigned char*)&gbStatusPacket[gbRxGetLength], gbRxPacketLength - gbRxGetLength );
169         gbRxGetLength += nRead;
170         if( gbRxGetLength < gbRxPacketLength )
171         {
172             gbCommStatus = COMM.RXWAITING;
173             return;
174         }
175     }
176
177     // Check checksum
178     for( i=0; i<(gbStatusPacket[LENGTH]+1); i++ )
179         checksum += gbStatusPacket[i+2];
180     checksum = ~checksum;
181
182     if( gbStatusPacket[gbStatusPacket[LENGTH]+3] != checksum )
183     {

```

```

184     gbCommStatus = COMMLRXCORRUPT;
185     giBusUsing = 0;
186     return;
187 }
188
189 gbCommStatus = COMMLRXSUCCESS;
190 giBusUsing = 0;
191 }
192
193 void dxl_txrx_packet(void)
194 {
195     dxl_tx_packet();
196     if( gbCommStatus != COMMLTXSUCCESS )
197         return;
198
199     do{
200         dxl_rx_packet();
201     }while( gbCommStatus == COMMLRXWAITING );
202 }
203
204
205 int dxl_get_result(void)
206 {
207     return gbCommStatus;
208 }
209
210 void dxl_set_txpacket_id( int id )
211 {
212     gbInstructionPacket[ID] = (unsigned char)id;
213 }
214
215 void dxl_set_txpacket_instruction( int instruction )
216 {
217     gbInstructionPacket[INSTRUCTION] = (unsigned char)instruction;
218 }
219
220 void dxl_set_txpacket_parameter( int index, int value )
221 {
222     gbInstructionPacket[PARAMETER+index] = (unsigned char)value;
223 }
224
225 void dxl_set_txpacket_length( int length )
226 {
227     gbInstructionPacket[LENGTH] = (unsigned char)length;
228 }
229
230 int dxl_get_rxpacket_error( int errbit )
231 {
232     if( gbStatusPacket[ERRBIT] & (unsigned char)errbit )
233         return 1;
234     return 0;
235 }
236
237
238 int dxl_get_rxpacket_length(void)
239 {
240     return (int)gbStatusPacket[LENGTH];
241 }
242
243 int dxl_get_rxpacket_parameter( int index )
244 {
245     return (int)gbStatusPacket[PARAMETER+index];
246 }
247
248 int dxl_makeword( int lowbyte, int highbyte )
249 {
250     unsigned short word;
251
252     word = highbyte;
253     word = word << 8;
254     word = word + lowbyte;
255     return (int)word;
256 }
257
258 int dxl_get_lowbyte( int word )
259 {
260     unsigned short temp;
261
262     temp = word & 0xff;
263     return (int)temp;
264 }
265
266 int dxl_get_highbyte( int word )
267 {
268     unsigned short temp;
269
270     temp = word & 0xff00;
271     temp = temp >> 8;
272     return (int)temp;
273 }
274
275 void dxl_ping( int id )

```

```

276 {
277     while (giBusUsing);
278
279     gbInstructionPacket[ID] = (unsigned char)id;
280     gbInstructionPacket[INSTRUCTION] = INST_PING;
281     gbInstructionPacket[LENGTH] = 2;
282
283     dxl_txrx_packet();
284 }
285
286 int dxl_read_byte( int id, int address )
287 {
288     while (giBusUsing);
289
290     gbInstructionPacket[ID] = (unsigned char)id;
291     gbInstructionPacket[INSTRUCTION] = INST_READ;
292     gbInstructionPacket[PARAMETER] = (unsigned char)address;
293     gbInstructionPacket[PARAMETER+1] = 1;
294     gbInstructionPacket[LENGTH] = 4;
295
296     dxl_txrx_packet();
297
298     return (int)gbStatusPacket[PARAMETER];
299 }
300
301 void dxl_write_byte( int id, int address, int value )
302 {
303     while (giBusUsing);
304
305     gbInstructionPacket[ID] = (unsigned char)id;
306     gbInstructionPacket[INSTRUCTION] = INST_WRITE;
307     gbInstructionPacket[PARAMETER] = (unsigned char)address;
308     gbInstructionPacket[PARAMETER+1] = (unsigned char)value;
309     gbInstructionPacket[LENGTH] = 4;
310
311     dxl_txrx_packet();
312 }
313
314 int dxl_read_word( int id, int address )
315 {
316     while (giBusUsing);
317
318     gbInstructionPacket[ID] = (unsigned char)id;
319     gbInstructionPacket[INSTRUCTION] = INST_READ;
320     gbInstructionPacket[PARAMETER] = (unsigned char)address;
321     gbInstructionPacket[PARAMETER+1] = 2;
322     gbInstructionPacket[LENGTH] = 4;
323
324     dxl_txrx_packet();
325
326     return dxl_makeword((int)gbStatusPacket[PARAMETER], (int)gbStatusPacket[PARAMETER+1]);
327 }
328
329 void dxl_write_word( int id, int address, int value )
330 {
331     while (giBusUsing);
332
333     gbInstructionPacket[ID] = (unsigned char)id;
334     gbInstructionPacket[INSTRUCTION] = INST_WRITE;
335     gbInstructionPacket[PARAMETER] = (unsigned char)address;
336     gbInstructionPacket[PARAMETER+1] = (unsigned char)dxl_get_lowbyte(value);
337     gbInstructionPacket[PARAMETER+2] = (unsigned char)dxl_get_highbyte(value);
338     gbInstructionPacket[LENGTH] = 5;
339
340     dxl_txrx_packet();
341 }

```

src/dynamixel.c

## 2.13 dxl\_hal.h

```

1 #ifndef _DYNAMIXEL_HAL_HEADER
2 #define _DYNAMIXEL_HAL_HEADER
3
4
5 #ifdef __cplusplus
6 extern "C" {
7 #endif
8
9
10 int dxl_hal_open(int deviceIndex, float baudrate);
11 void dxl_hal_close();
12 int dxl_hal_set_baud( float baudrate );
13 void dxl_hal_clear();

```



```

14 int dxl_hal_tx( unsigned char *pPacket, int numPacket );
15 int dxl_hal_rx( unsigned char *pPacket, int numPacket );
16 void dxl_hal_set_timeout( int NumRcvByte );
17 int dxl_hal_timeout();
18
19
20
21 #ifdef __cplusplus
22 }
23 #endif
24
25 #endif

```

src/dxl\_hal.h

## 2.14 dxl\_hal.c

```

1  #include <stdio.h>
2  #include <string.h>
3  #include <unistd.h>
4  #include <fcntl.h>
5  #include <termios.h>
6  #include <linux/serial.h>
7  #include <sys/ioctl.h>
8  #include <sys/time.h>
9
10 #include "dxl_hal.h"
11
12 int gSocket_fd = -1;
13 long glStartTime = 0;
14 float gfRcvWaitTime = 0.0f;
15 float gfByteTransTime = 0.0f;
16
17 char gDeviceName[20];
18
19 int dxl_hal_open(int deviceIndex, float baudrate)
20 {
21     struct termios newtio;
22     struct serial_struct serinfo;
23     char dev_name[100] = {0, };
24
25     sprintf(dev_name, "/dev/ttyUSB%d", deviceIndex);
26
27     strcpy(gDeviceName, dev_name);
28     memset(&newtio, 0, sizeof(newtio));
29     dxl_hal_close();
30
31     if((gSocket_fd = open(gDeviceName, O_RDWR|O_NOCTTY|O_NONBLOCK)) < 0) {
32         fprintf(stderr, "device open error: %s\n", dev_name);
33         goto DXL_HAL_OPEN_ERROR;
34     }
35
36     newtio.c_cflag = B38400|CS8|CLOCAL|CREAD;
37     newtio.c_iflag = IGNPAR;
38     newtio.c_oflag = 0;
39     newtio.c_lflag = 0;
40     newtio.c_cc[VMTIME] = 0; // time-out (TIME * 0.1 ) 0 : disable
41     newtio.c_cc[VMIN] = 0; // MIN read return
42
43     tcflush(gSocket_fd, TCIFLUSH);
44     tcsetattr(gSocket_fd, TCSANOW, &newtio);
45
46     if(gSocket_fd == -1)
47         return 0;
48
49     if(ioctl(gSocket_fd, TIOCGSERIAL, &serinfo) < 0) {
50         fprintf(stderr, "Cannot get serial info\n");
51         return 0;
52     }
53
54     serinfo.flags &= ~ASYNC_SPD_MASK;
55     serinfo.flags |= ASYNC_SPD_CUST;
56     serinfo.custom_divisor = serinfo.baud_base / baudrate;
57
58     if(ioctl(gSocket_fd, TIOCSSERIAL, &serinfo) < 0) {
59         fprintf(stderr, "Cannot set serial info\n");
60         return 0;
61     }
62
63     dxl_hal_close();
64
65     gfByteTransTime = (float)((1000.0f / baudrate) * 12.0f);
66
67     strcpy(gDeviceName, dev_name);

```

```

68  memset(&newtio, 0, sizeof(newtio));
69  dxl_hal_close();
70
71  if((gSocket_fd = open(gDeviceName, O_RDWR|O_NOCTTY|O_NONBLOCK)) < 0) {
72      fprintf(stderr, "device open error: %s\n", dev_name);
73      goto DXL_HAL_OPEN_ERROR;
74  }
75
76  newtio.c_cflag = B38400|CS8|CLOCAL|CREAD;
77  newtio.c_iflag = IGNPAR;
78  newtio.c_oflag = 0;
79  newtio.c_lflag = 0;
80  newtio.c_cc[VTIME] = 0; // time-out (TIME * 0.1 ) 0 : disable
81  newtio.c_cc[VMIN] = 0; // MIN read return
82
83  tcflush(gSocket_fd, TCIFLUSH);
84  tcsetattr(gSocket_fd, TCSANOW, &newtio);
85
86  return 1;
87
88 DXL_HAL_OPEN_ERROR:
89  dxl_hal_close();
90  return 0;
91 }
92
93 void dxl_hal_close()
94 {
95     if(gSocket_fd != -1)
96         close(gSocket_fd);
97     gSocket_fd = -1;
98 }
99
100 int dxl_hal_set_baud( float baudrate )
101 {
102     struct serial_struct serinfo;
103
104     if(gSocket_fd == -1)
105         return 0;
106
107     if(ioctl(gSocket_fd, TIOCGSERIAL, &serinfo) < 0) {
108         fprintf(stderr, "Cannot get serial info\n");
109         return 0;
110     }
111
112     serinfo.flags &= ~ASYNC_SPD_MASK;
113     serinfo.flags |= ASYNC_SPD_CUST;
114     serinfo.custom_divisor = serinfo.baud_base / baudrate;
115
116     if(ioctl(gSocket_fd, TIOCSSERIAL, &serinfo) < 0) {
117         fprintf(stderr, "Cannot set serial info\n");
118         return 0;
119     }
120
121     //dxl_hal_close();
122     //dxl_hal_open(gDeviceName, baudrate);
123
124     gfByteTransTime = (float)((1000.0f / baudrate) * 12.0f);
125     return 1;
126 }
127
128 void dxl_hal_clear(void)
129 {
130     tcflush(gSocket_fd, TCIFLUSH);
131 }
132
133 int dxl_hal_tx( unsigned char *pPacket, int numPacket )
134 {
135     return write(gSocket_fd, pPacket, numPacket);
136 }
137
138 int dxl_hal_rx( unsigned char *pPacket, int numPacket )
139 {
140     memset(pPacket, 0, numPacket);
141     return read(gSocket_fd, pPacket, numPacket);
142 }
143
144 static inline long myclock()
145 {
146     struct timeval tv;
147     gettimeofday (&tv, NULL);
148     return (tv.tv_sec * 1000 + tv.tv_usec / 1000);
149 }
150
151 void dxl_hal_set_timeout( int NumRcvByte )
152 {
153     glStartTime = myclock();
154     gfRcvWaitTime = (float)(gfByteTransTime*(float)NumRcvByte + 5.0f);
155 }
156
157 int dxl_hal_timeout(void)
158 {
159     long time;

```

```

160     time = myclock() - glStartTime;
161
162     if(time > gfRcvWaitTime)
163         return 1;
164     else if(time < 0)
165         glStartTime = myclock();
166
167     return 0;
168 }
169

```

src/dxl\_hal.c

## 2.15 communication.h

```

1  #ifndef COMMUNICATION_H
2  #define COMMUNICATION_H
3
4  int readWord(int, int);
5  int readByte(int, int);
6  int getResult();
7  int getRXpacketError(int);
8  void writeWord(int, int, int);
9  void writeByte(int, int, int);
10 void pingID(int);
11
12 #endif

```

include/communication.h

## 2.16 communication.cpp

```

1  #include <dynamixel.h>
2  #include <pthread.h>
3
4  //Mutex is used for multiple access from threads
5  //Best way would be to make communication atomic
6  //such that the communication would finish without
7  //being interrupted. That way could avoid timeout error
8  pthread_mutex_t mutex_comm = PTHREAD_MUTEX_INITIALIZER;
9
10 int readWord(int id, int address){
11     pthread_mutex_lock(&mutex_comm);
12     int temp = dxl_read_word(id, address);
13     pthread_mutex_unlock(&mutex_comm);
14     return temp;
15 }
16
17 int readByte(int id, int address){
18     pthread_mutex_lock(&mutex_comm);
19     int temp = dxl_read_byte(id, address);
20     pthread_mutex_unlock(&mutex_comm);
21     return temp;
22 }
23
24 int getResult(){
25     pthread_mutex_lock(&mutex_comm);
26     int temp = dxl_get_result();
27     pthread_mutex_unlock(&mutex_comm);
28     return temp;
29 }
30
31 int getRXpacketError(int errbit){
32     pthread_mutex_lock(&mutex_comm);
33     int temp = dxl_get_rxpacket_error(errbit);
34     pthread_mutex_unlock(&mutex_comm);
35     return temp;
36 }
37
38 void writeWord(int id, int address, int value){
39     pthread_mutex_lock(&mutex_comm);
40     dxl_write_word(id, address, value);
41     pthread_mutex_unlock(&mutex_comm);
42 }
43
44 void writeByte(int id, int address, int value){

```

```

45 pthread_mutex_lock( &mutex_comm );
46 dxl_write_byte(id, adress, value);
47 pthread_mutex_unlock( &mutex_comm );
48 }
49
50 void pingID(int id){
51 pthread_mutex_lock( &mutex_comm );
52 dxl_ping(id);
53 pthread_mutex_unlock( &mutex_comm );
54 }

```

src/communication.cpp

## 2.17 json\_processing.h

```

1
2 //defines
3 #define BUFFER_SIZE (256 * 1024) /* 256 KB */
4 #define URLFORMAT "https://wodinaz.com/%s"
5 #define URL_SIZE 256
6
7 //includes
8 #include <stdlib.h>
9 #include <string.h>
10 #include <stdio.h>
11 #include <string>
12 #include <vector>
13 #include <map>
14
15 using namespace std;
16
17 //functions
18 void json_test_function();
19 //example code that uses the four basic functions to communicate with the server
20
21 void debug_print_map(map<string, double> mymap);
22 // a debug function used to print maps received from the server
23
24 void debug_print_vector(vector<string> myvector);
25 //debug function used to print vectors
26
27
28 void json_send_data(map<string, double> mymap);
29 // Uploads the provided map of sensor values to the server
30
31 map<string, double> json_get_data(int id);
32 // Downloads sensor data from the server. The user must choose which agent (id) to receive from
33
34 void json_send_command(string cmd, int id);
35 // Uploads a command to the server.
36 //The agent with the corresponding id will download this command
37
38 vector<string> json_get_commands(int id);
39 //Download commands from the server.

```

include/json\_processing.h

## 2.18 json\_processing.cpp

```

1 /*
2  * Copyright (c) 2009–2013 Petri Lehtinen <petri@digip.org>
3  *
4  * Jansson is free software; you can redistribute it and/or modify
5  * it under the terms of the MIT license. See LICENSE for details.
6  */
7
8 #include <stdlib.h>
9 #include <string.h>
10 #include <stdio.h>
11
12 #include <jansson.h>
13
14 #include "http-functions.h"
15
16 #define BUFFER_SIZE (256 * 1024) /* 256 KB */
17

```

```

18 #define URLFORMAT      "https://wodinaz.com/%s"
19 #define URL_SIZE       256
20 int i=0;
21
22 //URL's
23 #define PATH.CONNECT    "connect"
24 #define PATH.DATA      "data/"
25 #define PATH.COMMAND   "command/"
26
27 //C++ stuff
28 #include <string>
29 #include <iostream>
30 #include <ostream>
31 #include <sstream>
32 #include <vector>
33 #include <map>
34 using namespace std;
35
36 int myID=0;
37 int testID=0;
38
39 //functions
40
41
42 void debug_print_map(map<string, double> mymap){
43     for (map<string, double>::iterator it=mymap.begin(); it!=mymap.end(); ++it)
44     {
45         string key = it->first;
46         double value = it->second;
47         printf ("sensor %s has value %f\n", key.c_str(), value);
48     }
49 }
50
51 void debug_print_vector(vector<string> myvector){
52     for (vector<string>::iterator it=myvector.begin(); it!=myvector.end(); ++it)
53     {
54         string command = *it;
55         printf ("command: %s\n", command.c_str());
56     }
57 }
58
59 string convertIntToString(int number)
60 {
61     if (number == 0)
62         return "0";
63     string temp="";
64     string returnvalue="";
65     while (number>0)
66     {
67         temp+=number%10+48;
68         number/=10;
69     }
70     for (int i=0; i<temp.length(); i++)
71         returnvalue+=temp[temp.length()-i-1];
72     return returnvalue;
73 }
74
75 int convertStringToInt(string inputString){
76     return atoi(inputString.c_str());
77 }
78
79 double convertStringToDouble(string inputString){
80     stringstream ss(inputString);
81     double result;
82     return ss >> result ? result : 0;
83 }
84
85 string convertDoubleToString(double number){
86     ostringstream convert; // stream used for the conversion
87     convert << number; // insert the textual representation of 'Number' in the characters in the stream
88     return convert.str(); // set 'Result' to the contents of the stream
89 }
90
91
92
93 map<string, double> json_get_data(int id){
94     printf("starting get_data\n");
95     map<string, double> data_map;
96     int root_length=0;
97     char *text_response;
98     char url[URL_SIZE];
99     string id_path=PATH.DATA;
100
101     string id_string = "client_" + convertIntToString(id);
102     id_path.append(id_string);
103     snprintf(url, URL_SIZE, URLFORMAT, id_path.c_str());
104     printf("url:%s\n", url);
105
106     text_response = http_request(url);
107     printf("response:%s\n", text_response);
108     json_t *root;
109     json_error_t error;

```

```

110 root = json_loads(text_response, 0, &error);
111 free(text_response);
112
113 if(!root)
114 {
115     fprintf(stderr, "error: on line %d: %s\n", error.line, error.text);
116     throw 202;
117 }
118
119 if(!json_is_array(root))
120 {
121     fprintf(stderr, "error: root is not an object\n");
122     json_decref(root);
123     root_length=1;
124 }
125
126 root_length=json_array_size(root);
127 printf("root_length:%d\n",root_length );
128 //getting the actual data
129 json_t *data, *time_stamp, *entry_id, *sensor, *sensor_value, *device_id;
130 double timeStamp,entryID,sensorValue, deviceID;
131 string sensor_name;
132 for (i=0;i<root_length;i++){ //DEBUG i<root_length
133     data = json_array_get(root, i);
134     if(!json_is_object(data))
135     {
136         fprintf(stderr, "error: commit data %d is not an object\n", i + 1);
137         json_decref(root);
138         throw 202;
139     }
140
141     time_stamp = json_object_get(data,"timestamp");
142     if (!json_is_string(time_stamp)){
143         printf("throwing jsonException\n");
144         throw 202;
145     }
146     else {
147
148         timeStamp = convertStringToDouble(json_string_value(time_stamp));
149         printf("timeStamp:%f\n",timeStamp );
150     }
151
152     entry_id = json_object_get(data, "_id");
153     if (!json_is_string(entry_id)){
154         printf("throwing jsonException\n");
155         throw 202;
156     }
157     else {
158         entryID =convertStringToDouble(json_string_value(entry_id));
159     }
160
161     sensor= json_object_get(data,"sensor");
162     if (!json_is_string(sensor)){
163         printf("throwing jsonException\n");
164         throw 202;
165     }
166     else {
167         sensor_name = json_string_value(sensor);
168         printf("sensor_name:%s\n",sensor_name.c_str() );
169     }
170
171     const char* snsr_name = sensor_name.c_str();
172     sensor_value = json_object_get(data,snsr_name);
173     if (!json_is_string(sensor_value)){
174         printf("throwing jsonException at sensor_value\n");
175         throw 202;
176     }
177     else {
178         sensorValue= convertStringToDouble(json_string_value(sensor_value));
179         printf("sensor_value:%f\n",sensorValue);
180     }
181
182     device_id = json_object_get(data,"device_id");
183     if (!json_is_string(device_id)){
184         printf("throwing jsonException at device id\n");
185         throw 202;
186     }
187     else {
188         deviceID = convertStringToDouble(json_string_value(device_id));
189         printf("deviceID:%f\n",deviceID);
190     }
191     //put stuff in returning map
192     data_map[sensor_name]=sensorValue;
193 }
194 return data_map;
195 }
196
197 void json_send_data(map<string, double> mymap){
198     //printf("starting send_data\n");
199
200     char url[URL_SIZE];
201

```

```

202
203 string id_string = convertIntToString(myID);
204 string http_path=PATH_DATA;
205 http_path.append("client_"+id_string);
206 string sensor_name;
207 string key;
208 double value;
209 string value_string;
210 string json_string;
211 for (map<string, double>::iterator it=mymap.begin(); it!=mymap.end(); ++it)
212 {
213     key = it->first;
214     value = it->second;
215     value_string=convertDoubleToString(value);
216     sensor_name=key;
217
218     string http_path=PATH_DATA;
219     http_path.append("client_"+id_string);
220     http_path.append("/");
221     http_path.append(sensor_name);
222     json_string="{ ";
223     json_string.append("\n");
224
225     json_string.append(sensor_name);
226     json_string.append("\n");
227     json_string.append(": ");
228     json_string.append("\n");
229     json_string.append("\n"+value_string+"\n"+"}");
230     snprintf(url, URL_SIZE, URL_FORMAT, http_path.c_str());
231     //printf("url:%s\n", url);
232     //printf("json_string:%s\n", json_string.c_str());
233
234     char *json_cstring = new char[json_string.length() + 1];
235     strcpy(json_cstring, json_string.c_str());
236     // do stuff
237
238     http_post(url, json_cstring);
239     free(json_cstring);
240 }
241
242
243 void json_send_command(string cmd, int id){
244     printf("starting send_commands\n");
245
246     char url[URL_SIZE];
247     string command=cmd;
248     string http_path=PATH_COMMAND;
249     string id_string = convertIntToString(id);
250     http_path.append("client_"+id_string);
251     string json_string;
252     http_path=PATH_COMMAND;
253     http_path.append("client_"+id_string);
254     json_string="{ ";
255     json_string.append("\n");
256
257     json_string.append("command");
258     json_string.append("\n");
259     json_string.append(": ");
260     json_string.append("\n");
261     json_string.append("\n"+command+"\n"+"}");
262     snprintf(url, URL_SIZE, URL_FORMAT, http_path.c_str());
263     printf("url:%s\n", url);
264     printf("json_string:%s\n", json_string.c_str());
265
266     char *json_cstring = new char[json_string.length() + 1];
267     strcpy(json_cstring, json_string.c_str());
268     // do stuff
269
270     http_post(url, json_cstring);
271     free(json_cstring);
272 }
273
274 vector<string> json_get_commands(int id){
275     //printf("starting get_commands\n");
276     vector<string> commands_vector;
277     int root_length=0;
278     char *text_response;
279     char url[URL_SIZE];
280     string id_path=PATH_COMMAND;
281
282     string id_string = "client_"+convertIntToString(id);
283     id_path.append(id_string);
284     snprintf(url, URL_SIZE, URL_FORMAT, id_path.c_str());
285     //printf("url:%s\n", url);
286
287     text_response = http_request(url);
288     //printf("response:%s\n", text_response);
289     json_t *root;
290     json_error_t error;
291     root = json_loads(text_response, 0, &error);
292     free(text_response);
293

```

```

294     if(!root)
295     {
296         fprintf(stderr, "error: on line %d: %s\n", error.line, error.text);
297         throw 202;
298     }
299
300     if(!json_is_array(root))
301     {
302         fprintf(stderr, "error: root is not an array\n");
303         json_decref(root);
304         root_length=1;
305     }
306
307     root_length=json_array_size(root);
308     //printf("root_length:%d\n",root_length );
309     //getting the actual data
310     json_t *data, *time_stamp, *iterator;
311     double timeStamp;
312     string command="";
313     for (i=0;i<root_length;i++){ //DEBUG i<root_length
314         data = json_array_get(root, i);
315         if (!json_is_object(data))
316         {
317             fprintf(stderr, "error: commit data %d is not an object\n", i + 1);
318             json_decref(root);
319             throw 202;
320         }
321
322         time_stamp = json_object_get(data,"timestamp");
323         if (!json_is_string(time_stamp)){
324             printf("throwing jsonException\n");
325             throw 202;
326         }
327         else {
328
329             timeStamp = convertStringToDouble(json_string_value(time_stamp));
330             printf("timeStamp:%f\n",timeStamp );
331         }
332         iterator =json_object_get(data,"command");
333         if (!json_is_string(iterator)){
334             printf("throwing jsonException\n");
335             throw 202;
336         }
337         else {
338             command = json_string_value(iterator);
339             //printf("command:%s\n",command.c_str());
340         }
341         commands_vector.push_back(command);
342     }
343     return commands_vector;
344 }
345
346
347 void json_test_function(){
348     map<string, double> debug_map;
349     debug_map["test1"]=8.9;
350     debug_map["test2"]=5678.456;
351     printf("Sending data\n");
352     json_send_data(debug_map);
353     printf("printing data\n");
354     debug_print_map(json_get_data(testID));
355
356
357     string command1="command one";
358     string command2="command two";
359     printf("sending commands\n");
360     json_send_command(command1,testID);
361     json_send_command(command2,testID);
362     printf("printing commands\n");
363     debug_print_vector(json_get_commands(testID));
364 }

```

src/json\_processing.cpp

## 2.19 http\_ functions.h

```

1 #ifndef HTTP_FUNCTIONS
2 #include <stdlib.h>
3 #include <string.h>
4 #include <stdio.h>
5
6 // make HTTP request to url
7 char* http_request(char *url);
8

```



```

9 //make a HTTP post to url
10 void http_post(char* url, char* json-string);
11 #endif

```

include/http\_functions.h

## 2.20 http\_functions.cpp

```

1
2 #include <stdlib.h>
3 #include <string.h>
4 #include <stdio.h>
5 #include <curl/curl.h>
6 using namespace std;
7
8 #define BUFFER_SIZE (256 * 1024) /* 256 KB */
9
10 #define URLFORMAT "https://wodinaz.com/%s"
11 #define URL_SIZE 256
12
13 struct write_result
14 {
15     char *data;
16     int pos;
17 };
18
19 static size_t write_response(void *ptr, size_t size, size_t nmemb, void *stream)
20 {
21     struct write_result *result = (struct write_result *)stream;
22
23     if(result->pos + size * nmemb >= BUFFER_SIZE - 1)
24     {
25         fprintf(stderr, "error: too small buffer\n");
26         return 0;
27     }
28
29     memcpy(result->data + result->pos, ptr, size * nmemb);
30     result->pos += size * nmemb;
31
32     return size * nmemb;
33 }
34
35 // make HTTP request to url
36 char* http_request(char *url)
37 {
38     CURL *curl = NULL;
39     CURLcode status;
40     struct curl_slist *headers = NULL;
41     char *data = NULL;
42     long code;
43
44     curl_global_init(CURL_GLOBAL_ALL);
45     curl = curl_easy_init();
46     if(!curl)
47         goto error;
48
49     data = (char*)malloc(BUFFER_SIZE);
50     if(!data)
51         goto error;
52
53     struct write_result write_result;
54     write_result.data=data;
55     write_result.pos=0;
56
57     curl_easy_setopt(curl, CURLOPT_URL, url);
58
59     curl_easy_setopt(curl, CURLOPT_HTTPHEADER, headers);
60
61     curl_easy_setopt(curl, CURLOPT_WRITEFUNCTION, write_response);
62     curl_easy_setopt(curl, CURLOPT_WRITEDATA, &write_result);
63
64     status = curl_easy_perform(curl);
65     if(status != 0)
66     {
67         fprintf(stderr, "error: unable to request data from %s:\n", url);
68         fprintf(stderr, "%s\n", curl_easy_strerror(status));
69         goto error;
70     }
71
72     curl_easy_getinfo(curl, CURLINFO_RESPONSE_CODE, &code);
73     if(code != 200)
74     {
75         fprintf(stderr, "error: server responded with code %ld\n", code);
76         goto error;
77     }
78 }

```

```

77     }
78
79     curl_easy_cleanup(curl);
80     curl_slist_free_all(headers);
81     curl_global_cleanup();
82
83     /* zero-terminate the result */
84     data[write_result.pos] = '\0';
85
86     return data;
87
88 error:
89     if (data)
90         free(data);
91     if (curl)
92         curl_easy_cleanup(curl);
93     if (headers)
94         curl_slist_free_all(headers);
95     curl_global_cleanup();
96     return NULL;
97 }
98
99 //post to server
100 void http_post(char* url, char* json_string){
101     CURL *curl;
102     CURLcode res;
103
104     /* In windows, this will init the winsock stuff */
105     curl_global_init(CURL_GLOBAL_ALL);
106     /* get a curl handle */
107     curl = curl_easy_init();
108     if (curl) {
109         /* First set the URL that is about to receive our POST. This URL can
110            just as well be a https:// URL if that is what should receive the
111            data. */
112         curl_easy_setopt(curl, CURLOPT_URL, url);
113
114         /* Now specify the POST data */
115         curl_easy_setopt(curl, CURLOPT_POSTFIELDS, json_string);
116
117         /* Perform the request, res will get the return code */
118         res = curl_easy_perform(curl);
119
120         /* Check for errors */
121         if (res != CURLE_OK)
122             fprintf(stderr, "curl_easy_perform() failed: %s\n",
123                 curl_easy_strerror(res));
124
125         //printf("return code:%d\n",res );
126         /* always cleanup */
127         curl_easy_cleanup(curl);
128     }
129     curl_global_cleanup();
130 }
131 }

```

src/http\_functions.cpp

# Chapter 3

## Example code

### 3.1 Car

```
1 #include <stdio.h>
2 #include <termio.h>
3 #include <unistd.h>
4 #include <dynamixel.h>
5 #include <time.h>
6 #include "car.h"
7
8 using namespace std;
9
10 //put ID of the wheels here
11 #define FRONT_RIGHT_WHEEL 1
12 #define BACK_RIGHT_WHEEL 3
13 #define FRONT_LEFT_WHEEL 0
14 #define BACK_LEFT_WHEEL 2
15
16
17 int main() {
18     int deviceIndex = 0;
19     int baudnum = 1;
20
21     printf("-----CAR TEST PROGRAM-----\n");
22
23     ////////// Open USB2Dynamixel //////////
24     if( dxl_initialize(deviceIndex, baudnum) == 0 )
25     {
26         printf( "Failed to open USB2Dynamixel!\n" );
27         printf( "Press Enter key to terminate...\n" );
28         getchar();
29         return 0;
30     }
31     else
32     {
33         printf( "Succeed to open USB2Dynamixel!\n" );
34
35         Car car1(FRONT_RIGHT_WHEEL, FRONT_LEFT_WHEEL, BACK_RIGHT_WHEEL, BACK_LEFT_WHEEL);
36         sleep(1);
37
38         car1.setSpeed(1023,1);
39         sleep(2);
40         car1.setSpeed(1023,0);
41         sleep(2);
42         car1.setSpeed(0,1);
43         sleep(2);
44
45         while(1)
46         {
47
48         }
49     }
50
51     // Close device
52     car1.setSpeed(0,1);
53     dxl_terminate();
54     return 0;
55 }
```

example/Car/src/main.cpp

## 3.2 Interface

```

1 #include <stdio.h>
2 #include <termio.h>
3 #include <unistd.h>
4 #include <dynamixel.h>
5 #include <time.h>
6 #include "car.h"
7 #include "manipulator.h"
8 #include "interface.h"
9
10 using namespace std;
11
12 //put ID of the wheels here
13 #define FRONT_RIGHT_WHEEL 1
14 #define BACK_RIGHT_WHEEL 3
15 #define FRONT_LEFT_WHEEL 0
16 #define BACK_LEFT_WHEEL 2
17
18 #define MAN_ONE 4 //zero at 511
19 #define MAN_TWO 7 //zero at 511, not allowed to go under
20 #define MAN_THREE 5 //zero at 511
21
22 #define GRIPPER_LEFT 12
23 #define GRIPPER_RIGHT 6
24
25 int main() {
26
27     int deviceIndex = 0;
28     int baudnum = 1;
29
30     printf("-----LOCAL INTERFACE TEST PROGRAM-----\n");
31
32     ////////// Open USB2Dynamixel //////////
33     if( dxl_initialize(deviceIndex, baudnum) == 0 )
34     {
35         printf( "Failed to open USB2Dynamixel!\n" );
36         printf( "Press Enter key to terminate...\n" );
37         getchar();
38         return 0;
39     }
40     else
41     {
42         printf( "Succeed to open USB2Dynamixel!\n" );
43
44         windowInit();
45         Car car1(FRONT_RIGHT_WHEEL, FRONT_LEFT_WHEEL, BACK_RIGHT_WHEEL, BACK_LEFT_WHEEL);
46         Manipulator manipulator1(MAN_ONE, MAN_TWO, MAN_THREE, GRIPPER_LEFT, GRIPPER_RIGHT);
47         sleep(1);
48
49         manipulator1.goToPosition(XSTART,YSTART,ZSTART);
50         manipulator1.setGripper(0);
51
52         while(1)
53         {
54             checkEvent(&manipulator1, &car1);
55         }
56
57
58
59
60     // Close device
61     car1.setSpeed(0,1);
62     dxl_terminate();
63     return 0;
64 }

```

example/Interface/src/main.cpp

## 3.3 Main

```

1 #include <stdio.h>

```

```

2 #include <termio.h>
3 #include <unistd.h>
4 #include <dynamixel.h>
5 #include <pthread.h>
6 #include <vector>
7 #include <string>
8 #include <time.h>
9 #include "car.h"
10 #include "manipulator.h"
11 #include "json_processing.h"
12 #include "sensor.h"
13
14 using namespace std;
15
16 //ID of wheels
17 #define FRONT_RIGHT_WHEEL 1
18 #define BACK_RIGHT_WHEEL 3
19 #define FRONT_LEFT_WHEEL 0
20 #define BACK_LEFT_WHEEL 2
21
22 //ID of manipulator arm
23 #define MAN_ONE 4 //zero at 511
24 #define MAN_TWO 7 //zero at 511, not allowed to go under
25 #define MAN_THREE 5 //zero at 511
26
27 //ID of gripper
28 #define GRIPPER_LEFT 12
29 #define GRIPPER_RIGHT 6
30
31 //ID of sensor
32 #define SENSOR 100
33
34 void *sendSensorData(void *ptr);
35
36 int main() {
37
38     pthread_t thread1;
39     int deviceIndex = 0;
40     int baudnum = 1;
41     string command;
42     vector<string> commands;
43     string strCheck = "position";
44
45     printf("-----MAIN PROGRAM-----\n");
46
47     ////////// Open USB2Dynamixel //////////
48     if( dxl_initialize(deviceIndex, baudnum) == 0 )
49     {
50         printf( "Failed to open USB2Dynamixel!\n" );
51         printf( "Press Enter key to terminate...\n" );
52         getchar();
53         return 0;
54     }
55     else
56         printf( "Succeed to open USB2Dynamixel!\n" );
57
58     Car car1(FRONT_RIGHT_WHEEL, FRONT_LEFT_WHEEL, BACK_RIGHT_WHEEL, BACK_LEFT_WHEEL);
59     Manipulator manipulator1(MAN_ONE, MAN_TWO, MAN_THREE, GRIPPER_LEFT, GRIPPER_RIGHT);
60     Sensor sensor1(SENSOR);
61     sleep(1);
62
63     sensor1.playMelody(6);
64     manipulator1.goToPosition(XSTART, YSTART, ZSTART);
65     manipulator1.setGripper(0);
66
67     //get old commands from server and disregard them
68     vector<string> dummy = json_get_commands(0);
69
70     //create thread for sending sensor data
71     pthread_create( &thread1, NULL, sendSensorData, &sensor1 );
72
73     while(1)
74     {
75
76         //get commands
77         while(commands.empty())
78         {
79             commands = json_get_commands(0);
80         }
81
82         //execute commands
83         while(!commands.empty())
84         {
85             command = commands.front();
86             commands.erase(commands.begin());
87             if(command == "forward")
88                 car1.setSpeed(1023,1);
89
90             else if(command == "backward")
91                 car1.setSpeed(1023,0);
92
93             else if(command == "stop")

```

```

94     car1.setSpeed(0,1);
95
96     else if(command == "leftTurn")
97         car1.turnCar(LEFT.TURN);
98
99     else if(command == "rightTurn")
100         car1.turnCar(RIGHT.TURN);
101
102     else if(command == "noTurn")
103         car1.turnCar(NO.TURN);
104
105     else if(command == "gripClose")
106         manipulator1.setGripper(1);
107
108     else if(command == "gripOpen")
109         manipulator1.setGripper(0);
110
111     else if(command.find(strCheck) != string::npos){
112         size_t found1 = command.find(" ");
113         size_t found2 = command.find(" ", found1+1);
114         size_t found3 = command.find(" ", found2+1);
115         string nr1 = command.substr(found1+1, found2-found1);
116         string nr2 = command.substr(found2+1, found3-found2);
117         string nr3 = command.substr(found3+1);
118
119         int x = atoi(nr1.c_str());
120         int y = atoi(nr2.c_str());
121         int z = atoi(nr3.c_str());
122         manipulator1.goToPosition(x, y, z);
123     }
124
125     else
126         printf("Unknown command\n");
127
128     printf("command: %s\n", command.c_str());
129 }
130
131 }
132
133
134
135 // Close device
136 car1.setSpeed(0,1);
137 dxl_terminate();
138 return 0;
139 }
140
141 //thread function for continuously sending data
142 void *sendSensorData(void *ptr){
143
144     //initialize sensor here?
145     Sensor* p = (Sensor*)ptr;
146     int data;
147     map<string,double> sensorData;
148     while(1){
149         //sleep for 100ms
150         sleep(1);
151
152         if(p->getMode() == FAILSAFE_MODE)
153         {
154             p->ping();
155             continue;
156         }
157
158         //get data and put it in the map
159         data = p->getIR(CENTER);
160         printf("\nIR center: %d\n",data);
161         sensorData["IR center"] = data;
162
163         data = p->getIR(LEFT);
164         printf("IR left: %d\n",data);
165         sensorData["IR left"] = data;
166
167         data = p->getIR(RIGHT);
168         printf("IR right: %d\n",data);
169         sensorData["IR right"] = data;
170         //send data
171         json_send_data(sensorData);
172         //clear map
173         sensorData.clear();
174     }
175     return NULL;
176 }

```

example/Main/src/main.cpp

## 3.4 Manipulator

```

1 #include <stdio.h>
2 #include <termio.h>
3 #include <unistd.h>
4 #include <dynamixel.h>
5 #include <time.h>
6 #include "manipulator.h"
7
8 using namespace std;
9
10 #define MAN_ONE      4    //zero at 511
11 #define MAN_TWO      7    //zero at 511, not allowed to go under
12 #define MAN_THREE    5    //zero at 511
13
14 #define GRIPPER_LEFT  12
15 #define GRIPPER_RIGHT 6
16
17 int main(){
18
19     int deviceIndex = 0;
20     int baudnum = 1;
21
22     printf("-----MANIPULATOR TEST PROGRAM-----\n");
23
24     ////////// Open USB2Dynamixel //////////
25     if( dxl_initialize(deviceIndex, baudnum) == 0 )
26     {
27         printf( "Failed to open USB2Dynamixel!\n" );
28         printf( "Press Enter key to terminate...\n" );
29         getchar();
30         return 0;
31     }
32     else
33         printf( "Succeed to open USB2Dynamixel!\n" );
34
35     Manipulator manipulator1(MAN_ONE, MAN_TWO, MAN_THREE, GRIPPER_LEFT, GRIPPER_RIGHT);
36     sleep(1);
37
38     manipulator1.setGripper(0);
39
40     //test drawing
41     // manipulator1.setGripper(1);
42     // manipulator1.drawLine(50,200,50,150,0);
43     // manipulator1.drawLine(50,175,25,175,0);
44     // manipulator1.drawLine(25,200,25,150,0);
45
46     while(1)
47     {
48
49         for(int i = 0; i < 130; i+=1)
50         {
51             manipulator1.goToPosition(0,170,i);
52             usleep(5000);
53         }
54         for(int i = 130; i > 0; i-=1)
55         {
56             manipulator1.goToPosition(0,170,i);
57             usleep(5000);
58         }
59
60         for(int i = 0; i < 100; i+=1)
61         {
62             manipulator1.goToPosition(i,170,0);
63             usleep(5000);
64         }
65         for(int i = 100; i > -100; i-=1)
66         {
67             manipulator1.goToPosition(i,170,0);
68             usleep(5000);
69         }
70         for(int i = -100; i < 0; i+=1)
71         {
72             manipulator1.goToPosition(i,170,0);
73             usleep(5000);
74         }
75     }
76
77
78
79     // Close device
80     dxl_terminate();
81     return 0;
82 }
83

```

example/Manipulator/src/main.cpp

## 3.5 Motor

```

1  #include <stdio.h>
2  #include <termio.h>
3  #include <unistd.h>
4  #include <dynamixel.h>
5  #include "motor.h"
6
7  using namespace std;
8
9  #define MOTOR_ID    1
10
11 int main(){
12
13     bool b = 0;
14     int deviceIndex = 0;
15     int baudnum = 1;
16
17     printf("-----MOTOR TEST PROGRAM-----\n");
18
19     ////////// Open USB2Dynamixel //////////
20     if( dxl_initialize(deviceIndex, baudnum) == 0 )
21     {
22         printf( "Failed to open USB2Dynamixel!\n" );
23         printf( "Press Enter key to terminate...\n" );
24         getchar();
25         return 0;
26     }
27     else
28         printf( "Succeed to open USB2Dynamixel!\n" );
29
30     Motor motor1(MOTOR_ID, SERVOMODE);
31
32     while(1)
33     {
34         try{
35             printf( "Press Enter key to continue!(press ESC and Enter to quit)\n" );
36             if(getchar() == 0x1b)
37                 break;
38
39             if(b){
40                 printf("motor1 to 300 degrees\n");
41                 motor1.setGoalPosition(1023);
42             }
43
44             else{
45                 printf("motor1 to 30 degrees\n");
46                 motor1.setGoalPosition(0);
47             }
48
49
50             b ^= 1;    //change b
51         }
52         catch(MotorException e){
53             printf("ID: %d lost\n",e.ID);
54             printError(e.status);
55             break;
56         }
57     }
58
59 }
60
61
62 // Close device
63 dxl_terminate();
64 return 0;
65 }
66

```

example/Motor/src/main.cpp

## 3.6 ReadWrite

```

1  //#####
2  //##          R O B O T I S          ##
3  //##          ReadWrite Example code for Dynamixel.      ##
4  //##          2009.11.10          ##
5  //#####
6  #include <stdio.h>
7  #include <termio.h>
8  #include <unistd.h>
9  #include <dynamixel.h>

```



```

10
11 // Control table address
12 #define P_GOAL_POSITION_L 30
13 #define P_GOAL_POSITION_H 31
14 #define P_PRESENT_POSITION_L 36
15 #define P_PRESENT_POSITION_H 37
16 #define P_MOVING 46
17
18 // Defulat setting
19 #define DEFAULT_BAUDNUM 1 // 1Mbps
20 #define DEFAULT_ID 1
21
22 void PrintCommStatus(int CommStatus);
23 void PrintErrorCode(void);
24
25 int main()
26 {
27     int baudnum = 1;
28     int GoalPos[2] = {0, 1023};
29     //int GoalPos[2] = {0, 4095}; // for Ex series
30     int index = 0;
31     int deviceIndex = 0;
32     int Moving, PresentPos;
33     int CommStatus;
34
35     printf( "\n\nRead/Write example for Linux\n\n" );
36     // Open USB2Dynamixel //
37     if( dxl_initialize(deviceIndex, baudnum) == 0 )
38     {
39         printf( "Failed to open USB2Dynamixel!\n" );
40         printf( "Press Enter key to terminate...\n" );
41         getchar();
42         return 0;
43     }
44     else
45     {
46         printf( "Succeed to open USB2Dynamixel!\n" );
47
48         while(1)
49         {
50             printf( "Press Enter key to continue!(press ESC and Enter to quit)\n" );
51             if(getchar() == 0x1b)
52                 break;
53
54             // Write goal position
55             dxl_write_word( DEFAULT_ID, P_GOAL_POSITION_L, GoalPos[index] );
56             do
57             {
58                 // Read present position
59                 PresentPos = dxl_read_word( DEFAULT_ID, P_PRESENT_POSITION_L );
60                 CommStatus = dxl_get_result();
61
62                 if( CommStatus == COMM_RXSUCCESS )
63                 {
64                     printf( "%d %d\n", GoalPos[index], PresentPos );
65                     PrintErrorCode();
66                 }
67                 else
68                 {
69                     PrintCommStatus(CommStatus);
70                     break;
71                 }
72
73                 // Check moving done
74                 Moving = dxl_read_byte( DEFAULT_ID, P_MOVING );
75                 CommStatus = dxl_get_result();
76                 if( CommStatus == COMM_RXSUCCESS )
77                 {
78                     if( Moving == 0 )
79                     {
80                         // Change goal position
81                         if( index == 0 )
82                             index = 1;
83                         else
84                             index = 0;
85                     }
86                     PrintErrorCode();
87                 }
88                 else
89                 {
90                     PrintCommStatus(CommStatus);
91                     break;
92                 }
93             } while( Moving == 1 );
94         }
95
96         // Close device
97         dxl_terminate();
98         printf( "Press Enter key to terminate...\n" );
99         getchar();
100         return 0;
101 }

```

```

102 // Print communication result
103 void PrintCommStatus(int CommStatus)
104 {
105     switch (CommStatus)
106     {
107     case COMM.TXFAIL:
108         printf("COMM.TXFAIL: Failed transmit instruction packet!\n");
109         break;
110
111     case COMM.TXERROR:
112         printf("COMM.TXERROR: Incorrect instruction packet!\n");
113         break;
114
115     case COMM.RXFAIL:
116         printf("COMM.RXFAIL: Failed get status packet from device!\n");
117         break;
118
119     case COMM.RXWAITING:
120         printf("COMM.RXWAITING: Now recieving status packet!\n");
121         break;
122
123     case COMM.RXTIMEOUT:
124         printf("COMM.RXTIMEOUT: There is no status packet!\n");
125         break;
126
127     case COMM.RXCORRUPT:
128         printf("COMM.RXCORRUPT: Incorrect status packet!\n");
129         break;
130
131     default:
132         printf("This is unknown error code!\n");
133         break;
134     }
135 }
136
137 // Print error bit of status packet
138 void PrintErrorCode()
139 {
140     if (dxl_get_rxpacket_error(ERRBIT_VOLTAGE) == 1)
141         printf("Input voltage error!\n");
142
143     if (dxl_get_rxpacket_error(ERRBIT_ANGLE) == 1)
144         printf("Angle limit error!\n");
145
146     if (dxl_get_rxpacket_error(ERRBIT_OVERHEAT) == 1)
147         printf("Overheat error!\n");
148
149     if (dxl_get_rxpacket_error(ERRBIT_RANGE) == 1)
150         printf("Out of range error!\n");
151
152     if (dxl_get_rxpacket_error(ERRBIT_CHECKSUM) == 1)
153         printf("Checksum error!\n");
154
155     if (dxl_get_rxpacket_error(ERRBIT_OVERLOAD) == 1)
156         printf("Overload error!\n");
157
158     if (dxl_get_rxpacket_error(ERRBIT_INSTRUCTION) == 1)
159         printf("Instruction code error!\n");
160 }

```

example/ReadWrite/ReadWrite.c

## 3.7 Sensor

```

1 #include <stdio.h>
2 #include <termio.h>
3 #include <unistd.h>
4 #include <dynamixel.h>
5 #include "sensor.h"
6 #include "songs.h"
7
8
9 using namespace std;
10
11 #define SENSOR    100
12
13 int main() {
14
15     int deviceIndex = 0;
16     int baudnum = 1;
17
18     printf("-----Sensor TEST PROGRAM-----\n");
19
20     ////////// Open USB2Dynamixel //////////

```

```

21 if( dxl_initialize(deviceIndex, baudnum) == 0 )
22 {
23     printf( "Failed to open USB2Dynamixel!\n" );
24     printf( "Press Enter key to terminate...\n" );
25     getchar();
26     return 0;
27 }
28 else
29     printf( "Succeed to open USB2Dynamixel!\n" );
30
31
32 Sensor sensor1(SENSOR);
33 sensor1.playMelody(FurElise, sizeof(FurElise));
34 //sensor1.playMelody(Sirene, sizeof(Sirene));
35 //sensor1.playMelody(6);
36
37 while(1)
38 {
39
40 }
41
42 // Close device
43 dxl_terminate();
44 return 0;
45 }

```

example/Sensor/src/main.cpp

## 3.8 SyncWrite

```

1  //#####
2  ///# R O B O T I S ##
3  ///# SyncWrite Example code for Dynamixel. ##
4  ///# 2009.11.10 ##
5  //#####
6  #include <stdio.h>
7  #include <unistd.h>
8  #include <math.h>
9  #include <termio.h>
10
11 #include <dynamixel.h>
12
13 #define PI 3.141592f
14 #define NUMACTUATOR 3
15
16 // Control table address
17 #define P_GOAL_POSITION_L 30
18 #define P_GOAL_POSITION_H 31
19 #define P_GOAL_SPEED_L 32
20 #define P_GOAL_SPEED_H 33
21
22 // Defulat setting
23 #define DEFAULT_BAUDNUM 1 // 1Mbps
24 #define NUMACTUATOR 3 // Number of actuator
25 #define STEP_THETA (PI / 100.0f) // Large value is more fast
26 #define CONTROL_PERIOD (10000) // usec (Large value is more slow)
27
28 void PrintCommStatus(int CommStatus);
29 void PrintErrorCode(void);
30
31 int main()
32 {
33     int id[NUMACTUATOR];
34     int baudnum = 1;
35     int deviceIndex = 0;
36     float phase[NUMACTUATOR];
37     float theta = 0;
38     int AmpPos = 512;
39     //int AmpPos = 2048; // for EX series
40     int GoalPos;
41     int i;
42     int CommStatus;
43     printf( "\n\nSyncWrite example for Linux\n\n" );
44
45     // Initialize id and phase
46     for( i=0; i<NUMACTUATOR; i++ )
47     {
48         id[i] = i+1;
49         phase[i] = 2*PI * (float)i / (float)NUMACTUATOR;
50     }
51
52     //##### Open USB2Dynamixel #####
53     if( dxl_initialize(deviceIndex, baudnum) == 0 )
54     {

```

```

55     printf( "Failed to open USB2Dynamixel!\n" );
56     printf( "Press Enter key to terminate...\n" );
57     getchar();
58     return 0;
59 }
60 else
61     printf( "Succeed to open USB2Dynamixel!\n" );
62
63 // Set goal speed
64 dxl_write_word( BROADCAST_ID, P_GOAL_SPEED_L, 0 );
65 // Set goal position
66 dxl_write_word( BROADCAST_ID, P_GOAL_POSITION_L, AmpPos );
67
68 while(1)
69 {
70     printf( "Press Enter key to continue!(press ESC and Enter to quit)\n" );
71     if( getchar() == 0x1b)
72         break;
73
74     theta = 0;
75     do
76     {
77         // Make syncwrite packet
78         dxl_set_txpacket_id( BROADCAST_ID );
79         dxl_set_txpacket_instruction( INST_SYNC_WRITE );
80         dxl_set_txpacket_parameter( 0, P_GOAL_POSITION_L );
81         dxl_set_txpacket_parameter( 1, 2 );
82         for( i=0; i<NUMACTUATOR; i++ )
83         {
84             dxl_set_txpacket_parameter( 2+3*i, id[i] );
85             GoalPos = (int)((sin(theta+phase[i]) + 1.0) * (double)AmpPos);
86             printf( "%d ", GoalPos );
87             dxl_set_txpacket_parameter( 2+3*i+1, dxl_get_lowbyte( GoalPos ) );
88             dxl_set_txpacket_parameter( 2+3*i+2, dxl_get_highbyte( GoalPos ) );
89         }
90         dxl_set_txpacket_length( (2+1)*NUMACTUATOR+4 );
91
92
93         printf( "\n" );
94
95         dxl_txrx_packet();
96         CommStatus = dxl_get_result();
97         if( CommStatus == COMM_RXSUCCESS )
98         {
99             PrintErrorCode();
100         }
101         else
102         {
103             PrintCommStatus(CommStatus);
104             break;
105         }
106
107         theta += STEP_THETA;
108         usleep( CONTROL_PERIOD );
109     } while( theta < 2*PI );
110 }
111
112 dxl_terminate();
113 printf( "Press Enter key to terminate...\n" );
114 getchar();
115
116 return 0;
117 }
118
119 // Print communication result
120 void PrintCommStatus( int CommStatus )
121 {
122     switch( CommStatus )
123     {
124     case COMM_TXFAIL:
125         printf( "COMM_TXFAIL: Failed transmit instruction packet!\n" );
126         break;
127
128     case COMM_TXERROR:
129         printf( "COMM_TXERROR: Incorrect instruction packet!\n" );
130         break;
131
132     case COMM_RXFAIL:
133         printf( "COMM_RXFAIL: Failed get status packet from device!\n" );
134         break;
135
136     case COMM_RXWAITING:
137         printf( "COMM_RXWAITING: Now recieving status packet!\n" );
138         break;
139
140     case COMM_RXTIMEOUT:
141         printf( "COMM_RXTIMEOUT: There is no status packet!\n" );
142         break;
143
144     case COMM_RXCORRUPT:
145         printf( "COMM_RXCORRUPT: Incorrect status packet!\n" );
146

```

```
147     break;
148
149 default:
150     printf("This is unknown error code!\n");
151     break;
152 }
153 }
154
155 // Print error bit of status packet
156 void PrintErrorCode()
157 {
158     if (dxl_get_rxpacket_error(ERRBIT_VOLTAGE) == 1)
159         printf("Input voltage error!\n");
160
161     if (dxl_get_rxpacket_error(ERRBIT_ANGLE) == 1)
162         printf("Angle limit error!\n");
163
164     if (dxl_get_rxpacket_error(ERRBIT_OVERHEAT) == 1)
165         printf("Overheat error!\n");
166
167     if (dxl_get_rxpacket_error(ERRBIT_RANGE) == 1)
168         printf("Out of range error!\n");
169
170     if (dxl_get_rxpacket_error(ERRBIT_CHECKSUM) == 1)
171         printf("Checksum error!\n");
172
173     if (dxl_get_rxpacket_error(ERRBIT_OVERLOAD) == 1)
174         printf("Overload error!\n");
175
176     if (dxl_get_rxpacket_error(ERRBIT_INSTRUCTION) == 1)
177         printf("Instruction code error!\n");
178 }
```

example/SyncWrite/SyncWrite.c



# Chapter 4

## Server code

### 4.1 Installation notes

```
1 Requirements:
2 MongoDB
3 Python
4     pip (http://www.pip-installer.org/en/latest/)
5     virtualenv (http://www.virtualenv.org/en/latest/)
6
7 Setup:
8 In this directory:
9 # virtualenv --no-site-packages venv
10 # source venv/bin/activate
11 # pip install -r requirements.txt
12 # python server/server.py
```

server/INSTALL

### 4.2 Utility functions

```
1 from flask import Response
2 from functools import wraps
3 from helpers import unicode_to_str
4
5 def get_str_object_or_404(action):
6     @wraps(action)
7     def wrapper(*args, **kwargs):
8         result = action(*args, **kwargs)
9         if not result:
10             return {}, 404, {'Access-Control-Allow-Origin': '*', 'Access-Control-Allow-Headers': 'accept, content-type, origin'}
11         else:
12             return unicode_to_str(result), 200, {'Access-Control-Allow-Origin': '*', 'Access-Control-Allow-Headers': 'accept, content-type, origin'}
13     return wrapper
```

server/tools/decorators.py

```
1 import time
2
3 def unicode_to_str(data):
4     if isinstance(data, dict):
5         ret = {}
6         for key, value in data.iteritems():
7             ret[unicode_to_str(key)] = unicode_to_str(value)
8         return ret
9     elif isinstance(data, list):
10         ret = []
11         for value in data:
12             ret.append(unicode_to_str(value))
```

```

13         return ret
14     else:
15         return str(data)
16
17 def get_microtime():
18     return int(round(time.time() * 1000))

```

server/tools/helpers.py

## 4.3 Server logic

```

1 from flask import request
2 from flask.ext import restful
3 from pymongo import MongoClient
4 from tools.decorators import get_str_object_or_404
5 from tools.helpers import get_microtime, unicode_to_str
6
7 mongodb = MongoClient().db
8
9 class OptionsResrouce(restful.Resource):
10     def options(self):
11         return {'Allow': 'GET,POST'}, 200, {'Access-Control-Allow-Origin': '*', 'Access-Control-Allow-Methods': 'POST,GET',
12             'Access-Control-Allow-Headers': 'accept, content-type, origin'}
13
14 class Status(restful.Resource):
15     def __init__(self):
16         self.collection = mongodb.status
17
18     @get_str_object_or_404
19     def get(self, id):
20         return self.collection.find_one({'device_id': id})
21
22     def post(self, id):
23         data = request.get_json(force=True, cache=False)
24         data["device_id"] = id
25         data["timestamp"] = get_microtime()
26
27         self.collection.update({'device_id': id}, data, upsert=True)
28
29         return {"commands": Command().get(id)}
30
31     def options(self, id):
32         return {'Allow': 'GET,POST'}, 200, {'Access-Control-Allow-Origin': '*', 'Access-Control-Allow-Methods': 'POST,GET',
33             'Access-Control-Allow-Headers': 'accept, content-type, origin'}
34
35 class StatusOptions(OptionsResrouce):
36     pass
37
38 class Data(restful.Resource):
39     def __init__(self):
40         self.collection = mongodb.data
41
42     @get_str_object_or_404
43     def get(self, id, sensor):
44         return self.collection.find_one({'device_id': id, 'sensor': sensor})
45
46     def post(self, id, sensor):
47         data = request.get_json(force=True, cache=False)
48
49         data["device_id"] = id
50         data["timestamp"] = get_microtime()
51         data["sensor"] = sensor
52         self.collection.update({'device_id': id, 'sensor': sensor}, data, upsert=True)
53
54         return {"commands": Command().get(id)}
55
56     def options(self, id):
57         return {'Allow': 'GET,POST'}, 200, {'Access-Control-Allow-Origin': '*', 'Access-Control-Allow-Methods': 'POST,GET',
58             'Access-Control-Allow-Headers': 'accept, content-type, origin'}
59
60 class DataOptions(OptionsResrouce):
61     pass
62
63 class Data_Collection(restful.Resource):
64     def __init__(self):
65         self.collection = mongodb.data
66
67     @get_str_object_or_404
68     def get(self, id):
69         return [sensor for sensor in self.collection.find({'device_id': id})]
70
71     def post(self, id):
72         data = request.get_json(force=True, cache=False)

```



```

71         for sensor_data in data:
72             sensor_data["device_id"] = id
73             sensor_data["timestamp"] = get_microtime()
74             self.collection.update({'device_id': id, 'sensor': sensor_data['sensor']}, sensor_data, upsert=True)
75
76         return {"commands": Command().get(id)}
77
78     def options(self, id):
79         return {'Allow': 'GET,POST', 200, {'Access-Control-Allow-Origin': '*', 'Access-Control-Allow-Methods': 'POST,GET',
80             'Access-Control-Allow-Headers': 'accept, content-type, origin '}}
81
82 class Command(restful.Resource):
83     def __init__(self):
84         self.collection = mongoddb.commands
85         self.id = hex(id(self))
86
87     def __get_document_lock(self, id):
88         document = self.collection.find_one({"device_id": id})
89
90         if not document:
91             self.collection.insert({"device_id": id, "state": self.id, "queue": []})
92             document = self.collection.find_one({"device_id": id})
93
94         while document["state"] != self.id:
95             while document["state"] != "ready":
96                 document = self.collection.find_one({"device_id": id})
97                 self.collection.update({"device_id": id, "state": "ready"}, {"$set": {"state": self.id}})
98                 document = self.collection.find_one({"device_id": id})
99
100     def __free_document_lock(self, id):
101         self.collection.update({"device_id": id}, {"$set": {"state": "ready"}})
102
103     def get(self, id):
104         self.__get_document_lock(id)
105
106         try:
107             document = self.collection.find_one({"device_id": id})
108             self.collection.update({"device_id": id}, {"$set": {"queue": []}})
109         finally:
110             self.__free_document_lock(id)
111
112         return unicode.to_str(document["queue"])
113
114     def post(self, id):
115         command = request.get_json(force=True, cache=False)
116         command["timestamp"] = get_microtime()
117
118         self.__get_document_lock(id)
119
120         try:
121             self.collection.update({"device_id": id}, {"$push": {"queue": command}})
122         finally:
123             self.__free_document_lock(id)
124
125         return {}, 200, {'Access-Control-Allow-Origin': '*', 'Access-Control-Allow-Headers': 'accept, content-type, origin '}}
126
127     def options(self, id):
128         return {'Allow': 'GET,POST', 200, {'Access-Control-Allow-Origin': '*', 'Access-Control-Allow-Methods': 'POST,GET',
129             'Access-Control-Allow-Headers': 'accept, content-type, origin '}}
130
131 class CommandOptions(OptionsResource):
132     pass

```

server/resources.py

## 4.4 Main program

```

1 from flask import Flask
2 from flask.ext import restful
3 import resources
4
5 app = Flask(__name__)
6 api = restful.Api(app)
7
8 api.add_resource(resources.Status, '/status/<string:id>')
9 api.add_resource(resources.StatusOptions, '/status')
10 api.add_resource(resources.Data, '/data/<string:id>/<string:sensor>')
11 api.add_resource(resources.DataOptions, '/data')
12 api.add_resource(resources.DataCollection, '/data/<string:id>')
13 api.add_resource(resources.Command, '/command/<string:id>')
14 api.add_resource(resources.CommandOptions, '/command')
15
16 if __name__ == '__main__':
17     app.run(debug=True)

```

---

server/server.py