An Implementation of Database on LNC Controller

工業4.0下之軟硬體系統開發
Final Project

黎純蕙*、林晨涵†、李宜修‡、李昱伯§
Prof. Meng-Shiun Tsai

M.E. , National Taiwan University
June 24, 2019

Contents

*R07522840, astralee95@gmail.com
†R07522843, cyanlin0104@gmail.com
‡R07522802, r07522802@ntu.edu.tw
§R07522803, albert40375042@gmail.com

1 Introduction

In this final project, we proposed a framework for linking controller and database. The figure below shows the full framework of the process. Detail process and information will be included in the following contents. All works can be cloned from github:

```
>> git clone https://github.com/AstraLee/finalproject_industry4.git
```

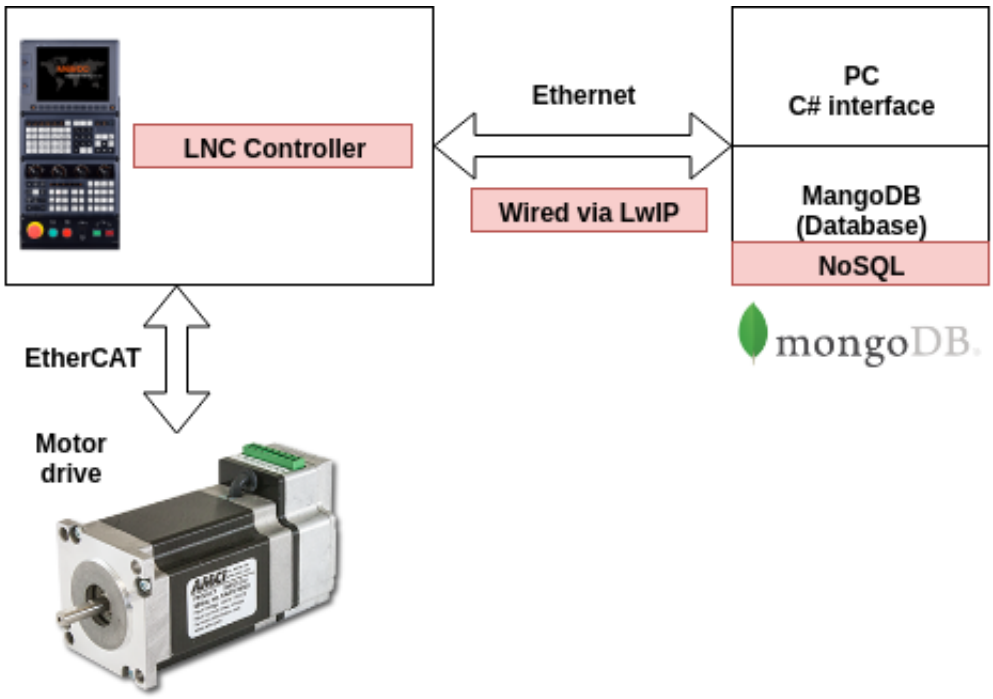


Figure 1: Framework of the experiment.

1.1 Scenario

Step 1: Setting up GUI for controller in C# (provided by LNC Tech.)

Step 2: Connecting to controller via wired Ethernet.

Step 3: Uploading G-code for controller in C# via File Transfer Protocol.

Step 4: Starting up motor drive by running uploaded G-code.

Step 5: Receiving encoder data and sending to server (.py) via socket.

Step 6: Listening to any client and ready to be connected.

Step 7: Creating MongoClient, connecting to server and receiving data simutaneously.

Step 8: Storing data into database and collection.

1.2 Hardware

1.2.1 LNC controller

LNC controllers are widely applied to all kinds of tool machines including turning machine, milling machine, tapping machine, grinding machine etc.[1]

- 10.4"TFT LCD.
- Control 9+6 axis.
- Support Mll/RTEX/ EtherCAT communication protocol.
- High-speed, high- precision and wiring saving.
- Provides interpolations to satisfy the requirements of high level of turn-milling.
- Various intelligent functions.
- Particle-button and support USB drive.

1.3 Software

1.3.1 SocketIO

A mechanism for allowing communication between processes where running on same or different computers connected on a network[2].

- Instant messaging and chat.
- Real-time analytics by pushing data to clients that get represented as real-time counters, charts or log.
- Documentation collaboration.

Initialization of server and setup server:

```
class SocketServer:
def __init__(self, host, port):
    self.port = port
    self.host = host
    self.server_socket = None
    self.client_socket = None
    self.recv_buffer = ""
def init_socket(self):
    self.server_socket = socket.socket(socket.AF_INET, socket.
 SOCK_STREAM)
    print("Server socket created")
```

[1] https://www.lnc.com.tw/en-us/products/651310d3-07e8-4065-96e2-e9cee323a966/t6800d-m(vertical_pbo)/mod_4b83b157-94cd-45ad-a36e-0a85483f4009

[2] https://socket.io/docs/using-multiple-nodes/

```
11
12          self.server_socket.bind((self.host, self.port))
13          print("Server socket bound with host {} port {}".format(
    self.host, self.port))
```

Listening to any connection from client:

```
1   class SocketServer:
2     def listen(self, num):
3       print('Listening...')
4       self.server_socket.listen(num)
5       self.client_socket, address = self.server_socket.accept()
    # accept new connection
6       print("Connection from: " + str(address))
```

Receiving data from client and end connection if no data is sent:

```
1   class SocketServer:
2     def receive(self, n_bytes):
3       self.recv_buffer = self.client_socket.recv(n_bytes)
4       print("{0} bytes data received ".format(len(self.
    recv_buffer)))
5       return len(self.recv_buffer)
6
7     def close_client(self):
8       self.client_socket.close()
```

1.3.2 MongoDB

MongoDB is an open-source database developed by MongoDB Inc. that's scalable and flexible. MongoDB stores data in JSON-like documents that can vary in structure[3].

- Easy to install and set up.
- Easy to map the document objects to application code.
- Highly scalable and available, and includes support for out-of-the-box replication.
- Flexible data models (Schema Free)
  - NoSQL databases more relaxed in structure of data
- Clusters of cheap commodity servers to manage the data and transaction volumes

Why MongoDB?

Pros 1: Simple queries.

Pros 2: Functionality provided applicable to most web applications.

[3] https://www.sitepoint.com/an-introduction-to-mongodb/

Pros 3: Easy and fast integration of data.

Pros 4: A BSON (a JSON-like format) to store data.

Pros 5: Free and open source.

Pros 6: Support MapReduce operations for condensing a large volume of data into useful aggregated results.

Cons 1: Not well suited for heavy and complex transactions systems.

Cons 2: NoSQL are still implementing their basic feature set

2 Basic Scenario

2.1 Ethernet Connection

TCPIP connection between LNC controller to PC.

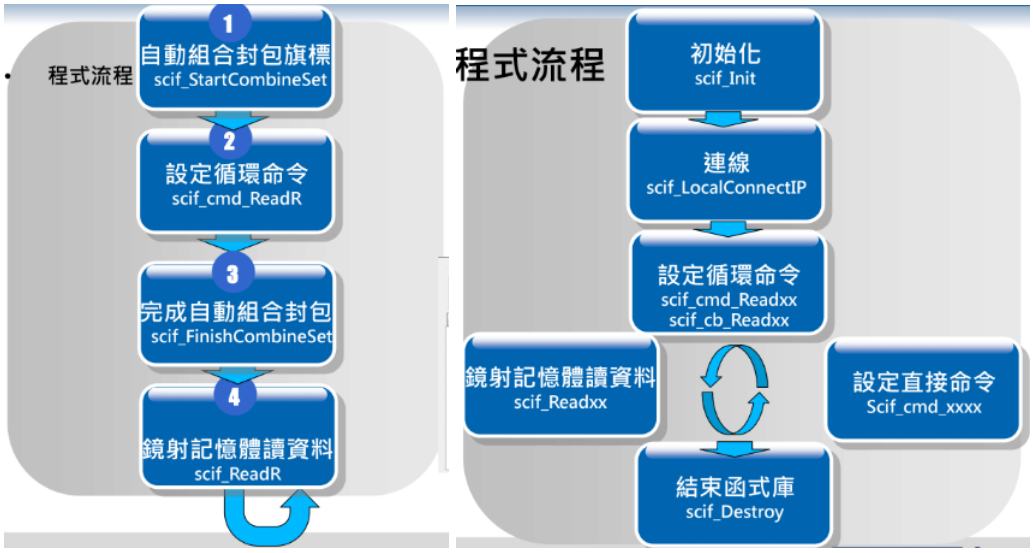


Figure 2: The pragramming scheme provided by LNC.

The connections on the C# interface:

- Method #1 use wired connection and detect IP address and connect.
- Method #2 use specific wifi configuration and connect to controller.
- Database connection allows data transmission to database (MongoDB).

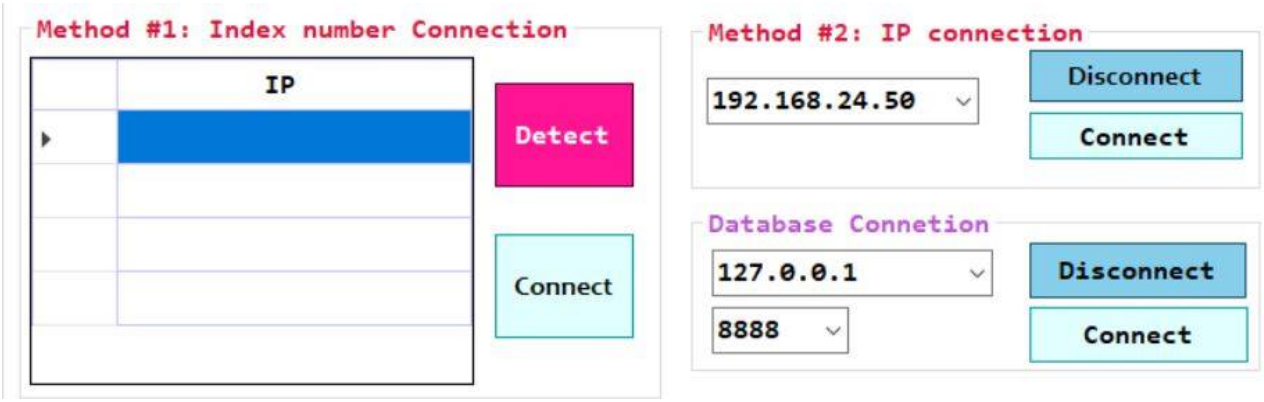


Figure 3: User Interface of Ethernet connection.

2.2 Data Extraction

- Extract data (eg. coordinates, loading, error) stored in R values.
- Get polling of the mirror memory for important R values.

```
private void ReadPos()
{
    for (int i = 0; i < 6; i++)
    {
        Pos_MAC[i] = scif_dll.scif_ReadR(scif_dll.R_AXIS_R_INT_MACHINE_POS + Convert.ToUInt32(i));
        Pos_ABS[i] = scif_dll.scif_ReadR(scif_dll.R_AXIS_R_INT_POS_ABSOLUTE + Convert.ToUInt32(i));
        Load[i] = scif_dll.scif_ReadR(scif_dll.R_LOAD + Convert.ToUInt32(i));
    }
}
```

```
public void FormCoorSetPolling()
{
    scif_dll.scif_cmd_ClearAll(scif_dll.SC_POLLING_CMD, ServerIdx);
    scif_dll.scif_StartCombineSet(ServerIdx);       //自動組合封包旗標
    scif_dll.scif_cmd_ReadR(scif_dll.SC_POLLING_CMD, ServerIdx, scif_dll.R_AXIS_R_INT_MACHINE_POS, 6);   //機械座標
    scif_dll.scif_cmd_ReadR(scif_dll.SC_POLLING_CMD, ServerIdx, scif_dll.R_AXIS_R_INT_POS_ABSOLUTE, 6); //程式座標
    scif_dll.scif_cmd_ReadR(scif_dll.SC_POLLING_CMD, ServerIdx, scif_dll.R_LOAD, 6);                    //loading
```

Figure 4: ReadPos() function for collecting data.

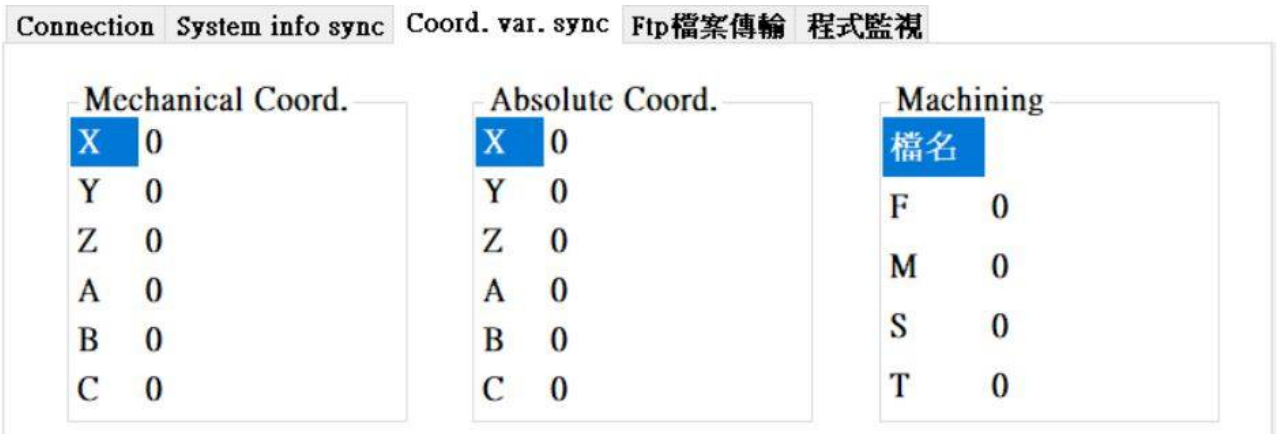


Figure 5: Data shown on the panel.

2.3 Controller

- Using G-code to control the hardware
- Using Ftp file transmission to upload G-code

Figure 6: Interface for uploading/downloading G-code files.

2.4 Machining Process

- Similar to typical factory application.
- G-code allows users to select machining process.
- Two R Values for G-code:
  - $\Phi R290100$: Return to **0** when machining ends, set to **1** to start working
  - $\Phi R290101$: Set **0** to go straight path to desired points; **1** to go curve path; **2** to stop machining

```
W_REG[290100,0]
WHILE[1]
    WHILE[1]
        #1=R_REG[290100]
        IF[#1==0]

        ELSE
            EXIT_WHILE
        END_IF
    END_WHILE
    #1=R_REG[290101]
    SELECT[#1]
    CASE 0:
        G01 X-500.000 Y500 F2500.000
        G04 X0.1
        G01 X0.000 Y-500 F2500.000
        G04 X0.1
        G01 Y0 F2500.000
    CASE 1:
        G01 S1 X-500.000 Y500 F2500.000
        G04 X0.1
        G01 S1 X0.000 Y-500 F2500.000
        G04 X0.1
        G01 S1 X0.000 Y0 F2500.000
    CASE 2:
        EXIT_WHILE
    END_SELECT
    W_REG[290100,0]
END_WHILE
M30;
```

Figure 7: Typical G-code for machining.

Write a value ($\Phi R290100$) to G-code function:

```
W_REG[290100,0]
```

Read a value ($\Phi R290101$) from controller:

```
R_REG[290101]
```

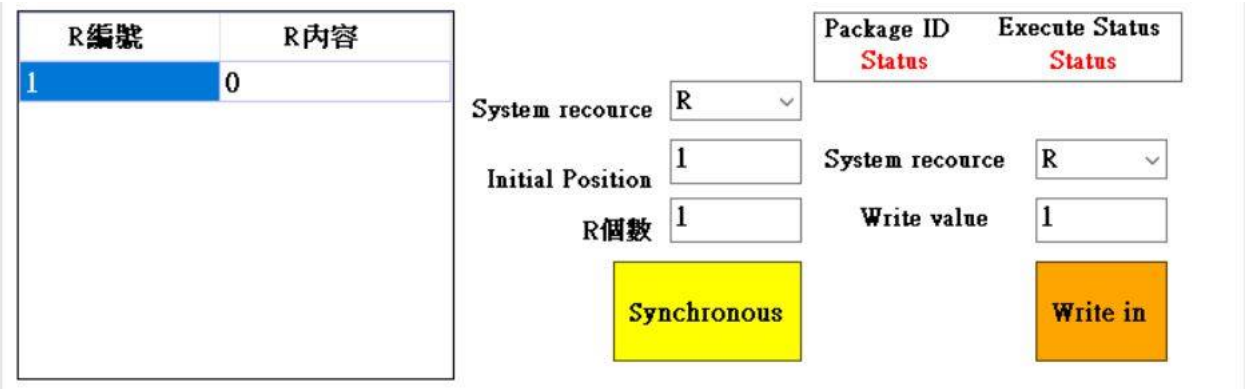


Figure 8: Interface to write R values to G-code.

2.5 Data Transmission

- LNC controller API function to Read R register value from Memory.
- **Socket** client(TCP) is created in same C# program.
- Socket client connects to local host (127.0.0.1).
- Waiting for server's acceptance/response.
- Sending Position and Payload data to socket server every second counts.
- Create a python socket server and **listen** to any connection.
- Socket server **connect**s to local host (127.0.0.1).
- Receive (**recv**) data from client.
- **PyMongoClient** gets data from server also on local host.
- Database is created simutaneously.

Database setup:

```
    connection = pymongo.MongoClient("mongodb://localhost:27017")

    database = connection['my_database']
    collection_ABS = database['my_collection_ABS']
    collection_MAC = database['my_collection_MAC']

    collection_LOAD = database['my_collection_LOAD']
    list_LOAD = ['LOAD_X', 'LOAD_Y']
    list_MAC = ['MAC_X', 'MAC_Y']
    list_ABS = ['ABS_X', 'ABS_Y']

    n_bytes = 3* NUM_AXIS * SIZE_INT

```

Use **struct.unpack** to decode the reverse Hex digits:

```
    while True:

    iResult = server.receive(n_bytes)
    timestamp = datetime.now().strftime('%Y-%m-%d %H:%M:%S')

    if(iResult > 0):
        X_MAC = struct.unpack('l', server.recv_buffer[0:4])[0]
        Y_MAC = struct.unpack('l', server.recv_buffer[4:8])[0]
        X_ABS = struct.unpack('l', server.recv_buffer[24:28])[0]
        Y_ABS = struct.unpack('l', server.recv_buffer[28:32])[0]
        X_Load = struct.unpack('l', server.recv_buffer[48:52])[0]
        Y_Load = struct.unpack('l', server.recv_buffer[52:56])[0]

        document_MAC = collection_MAC.insert_one({'timestamp' :
 timestamp, list_MAC[0]: X_MAC, list_MAC[1]: Y_MAC})
```

```
15          document_ABS = collection_ABS.insert_one({'timestamp' :
    timestamp, list_ABS[0]: X_ABS, list_ABS[1]: Y_ABS})
16          document_LOAD = collection_LOAD.insert_one({'timestamp' :
     timestamp, list_LOAD[0]: X_Load, list_LOAD[1]: Y_Load})
17
```

3 Application Scenario

Initially, we use R value of **Payload** as our main focus of optimization control. Therefore, the acceleration/deceleration of escalators reminds us of one of the key of the motion is depending on the magnitude of payload it endures. As to realize this scenario, we've extended the basic scenario futhermore.

Step #1 : Setting up GUI for controller in C# (provided by LNC Tech.)

Step #2 : Connecting to controller via wired Ethernet.

Step #3 : Uploading G-code for controller in C# via File Transfer Protocol.

> G-code reads loading R values and controls rotating speed.

Step #4 : Starting up motor drive by running uploaded G-code.

Step #5 : Receiving encoder (& payload) data and sending to server (.py) via socket.

Step #6 : Listening to any client to be connected.

Step #7 : Creating MongoClient, connecting to server and receiving data simutaneously.

> Dynamically updating plot in matplotlib.

Step #8 : Storing data into database and collection.

> Find data from collection w/ or w/o query.

3.1 G-code Payload R values

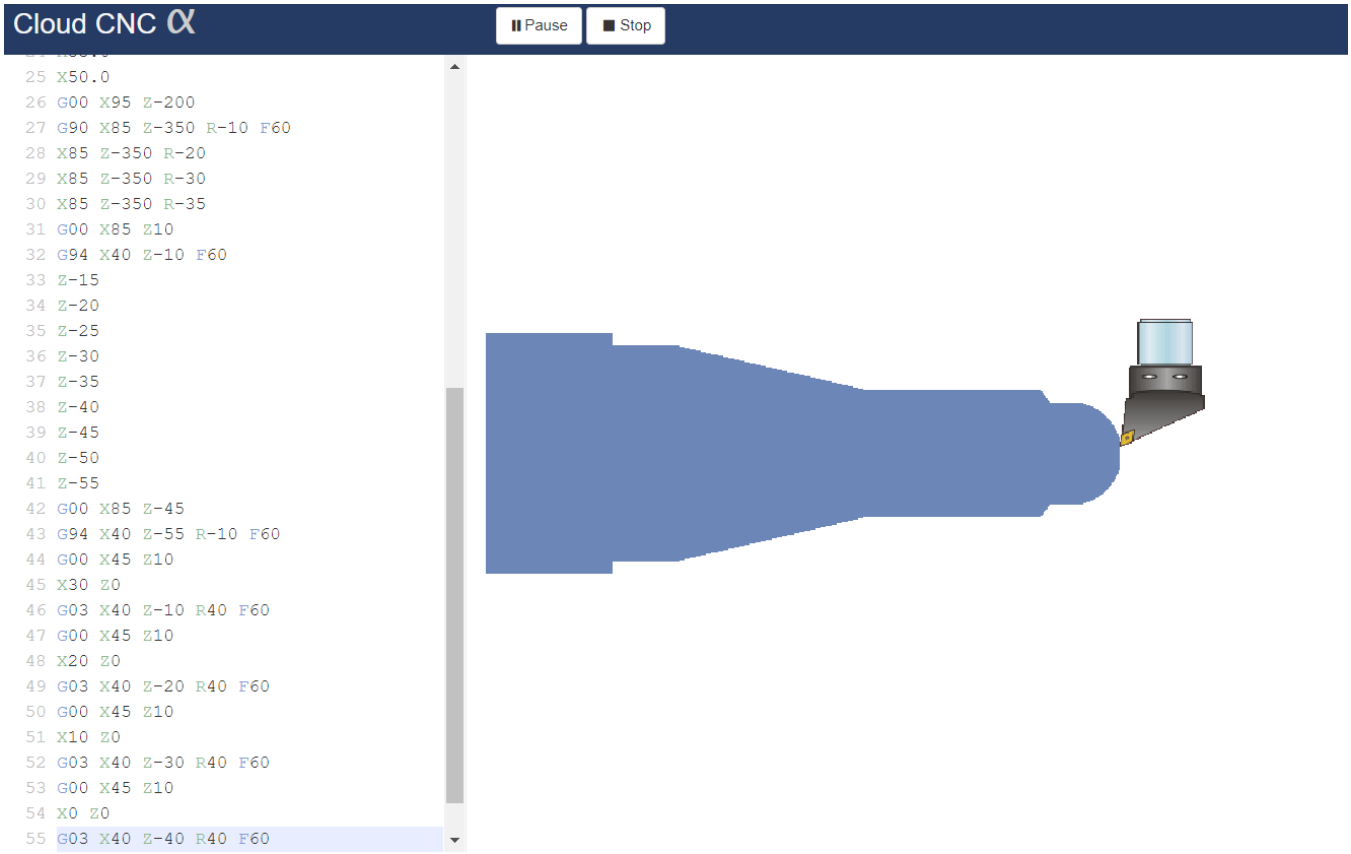
Designing G-code for the controller. Reading R value from controller where the loading is defined $\Phi R250096$ for x-axis, while $\Phi R250097$ is for y-axis. If the loading data is larger than 10, the speed will increase by 15, otherwise, increase by minor value 0.5.

```
W_REG[290100,0]
#2=0
WHILE[1]
    #1=R_REG[250096]

    IF[#1>10]
        #2=#2+15
        G01 X#2 Y#2 F8000.000
    ELSE
        #2=#2+0.5
        G01 X#2 Y#2 F2500.000
    END_IF
END_WHILE
M30;
```

Figure 9: G-code for application(ac/deceleration).

G-code simulation on CNC-α website[4].



[4]https://cnc-lathe-simulator.appspot.com/

3.2 Dynamically Updating Plot

Use **Matplotlib**, update figure and axis range once receiving data from database dynamically.

```
def Update_figure(idx, new_data):
  axes.set_xlim(0,idx)
  plot_data_x.append(idx)
  plot_data_y.append(new_data)
  line.set_xdata(plot_data_x)
  line.set_ydata(plot_data_y)
  plt.draw()
  plt.pause(1e-17)
```

3.3 Find data

From another point of view, once the database is built, other clients can access to the database and look for data. For our case, we hope that user can find data by giving some information of time, and investigate the process situation during that time interval.

Connect to MongoDB and find databases and collections:

```
connection = pymongo.MongoClient("mongodb://localhost:27017")
database = connection['my_database']
collection_ABS = database['my_collection_ABS']
collection_MAC = database['my_collection_MAC']
```

Giving a condition:

```
condition = {'timestamp': {'$gt':datetime(2019, 6, 15, 15, 8,
 33).strftime('%Y-%m-%d %H:%M:%S')}}
```

Showing the results in pandas form:

```
cursor = collection_ABS.find(condition)
dataFrame = pd.DataFrame(list(cursor))
```

Result:

```
In [3]: import time
        import pymongo
        import struct
        from struct import unpack
        from datetime import datetime
        import pandas as pd

In [4]: connection = pymongo.MongoClient("mongodb://localhost:27017")

        database = connection['my_database']
        collection_ABS = database['my_collection_ABS']
        collection_MAC = database['my_collection_MAC']

In [15]: condition = {'timestamp': {'$gt':datetime(2019, 6, 17, 18, 35, 33).strftime('%Y-%m-%d %H:%M:%S')}}

In [16]: cursor = collection_ABS.find(condition)
         dataFrame = pd.DataFrame(list(cursor))
         dataFrame

Out[16]:
```

	ABS_X	ABS_Y	_id	timestamp
0	1547941	1547941	5d076cf6f029e17e8bb82634	2019-06-17 18:35:34
1	1558504	1558504	5d076cf7f029e17e8bb82637	2019-06-17 18:35:35

4 Demo video

- Basic scenario - `https://youtu.be/4DcAY5w4KN0`
- Application scenario - `https://youtu.be/hwyQik2PvJg`

5 Conclusion

5.1 Discussions

- By Socket TCP, we send dataFrame from C# interface to python server and bind local Host as IP address.
- Simulating the process can provide graphical interface for users monitor the entire machining process easily.
- Socket sends data stream in Bytes array. Thus, we take advantages of Concept of Union (The same memory address) and transform **Int** to **Bytes** array (1 Int = 4 Bytes in C#) while receiver decodes Bytes array back into Int.
- G-code definition is slightly different to other industrial robot arm. Therefore, the implementation has to follow the predefined G function.
- The inverse kinematic for DoF less than 5 is simple, just $[\theta_1, \theta_2, ...]$ because the end-effector's axes are all decoupled and independent to each other. But once meet the 5-axis machining tool, inverse kinematic has to be done.

5.2 Conclusion

Through C# GUI to give commands to controller, and store data into the database at the same time is achievable. Data visualization and its noSQL structure can better improve data storage and management. Those applications can greatly optimize control in different scenarios.

Appendix

分工：
Database, server, client setup: 晨涵、純蕙
Controller, G-code: 宜修、昱伯