Turin-Robotics-RemoteControl-2019-06-20

CTI One Corporation

Version: 1.0
Date: June 20, 2019
Project Lead: Harry Li, Ph.D.

Team members: Shuwen Zheng, Prashanth Rajasekar

June-20-2019 TCP/IP Communication

Using TCP/IP to connect with robotics.

Connecting code:

```
import telnetlib

HOST = '192.168.1.148'

PORT = 8527

tn = telnetlib.Telnet(HOST, PORT, timeout = 1)

print("\nConnected : \n", tn)
```

June-20-2019 Login

Entering Username and password to control the robotics.

Login code:

```
login_str = str('<Bodys> <Cmd Name="Login" Status="Send"><Param
UserName = "administrator" Password="12345678"/></Cmd> </Bodys>')

print(login_str)

print("Authentication Successful")

tn.write((login_str+"\n").encode('ascii'))

print("Logged In to the TURIN System\n")

response = tn.read_until(b'/Bodys')
```

Notes:
The controller command serve command packet format is XML:

```
<Bodys>
	<Cmd Name=”XXX”
Status=”Send”/>
</Bodys>
```

we need to string XML in python code

June-20-2019 Joint rotation range

In joint system, each joint has a value representing its location.
Each of joint has limitation of their rotation.

J1	:	-100	<->	+100
J2	:	-60	<->	+135
J3	:	-80	<->	+80
J4	:	-160	<->	+160
J5	:	-135	<->	+135
J6	:	-360	<->	+369

ATTENTION:
DO NOT EXCEED LIMITATION.

June-20-2019 Sending command samples

After connecting robotics, we can use number to send preset command to robotics to test program.

```
while True:
    print("\nAvailable Tasks:")
    print("1. Trajectory1")
    print("Enter Task No: ")
    task=int(input())

    if task==1:
    action_str = str('<Bodys><Cmd Name="MotionEnd" Status="Send"/></Bodys>')  # stop motion
    tn.write((action_str+"\n").encode('ascii'))
    response = tn.read_until(b'/Bodys')
    time.sleep(1)
    action_str_1 = str('<Bodys> <Cmd Name="MotionBegin" Status="Send"> <Param FileName =
"1111.txt" StartLine = "0"/> </Cmd> </Bodys>') # Start motion, open a file    tn.write((action_str+"\
n").encode('ascii'))
    print("\nTask1 sent.\n")
    response = tn.read_until(b'/Bodys')
```

June-20-2019 command packet format

Following are different command to robotics:

1) Login command

```
<Bodys>
    <Cmd Name="Login"
Status="Send"><Param UserName =
"administrator" Password="12345678"/
></Cmd>
</Bodys>
```

2) Logout command

```
<Bodys>
    <Cmd Name="Logout" Status="Send"/>
</Bodys>
```

3) GetCurPos command (return the current world coordinate of the robot)

```
<Bodys>
    <Cmd Name="GetCurPos"
Status="Send"/>
</Bodys>
```

4) GetMotionStatus command (get the robot running status)

```
<Bodys>
    <Cmd Name="GetCurPos"
Status="Send"/>
</Bodys>
```

June-20-2019 command packet format

5) MotionBegin command (which is used to start the motion by opening a file)

```
<Bodys>
    <Cmd Name="MotionBegin"
Status="Send"><Param FileName =
”XXX.txt" StartLine = "0"/>
    </Cmd>
</Bodys>
```

6) MotionBegin command (which is used to start the motion by sending position)

```
<Bodys>
    <Cmd Name="MotionBegin"
Status="Send"><Param FileName =
”XXX.txt" StartLine = "0"/>
     <Data>MoveL(xxxx)</Data>
     </Cmd>
</Bodys>
```

7) MotionEnd command (stop motion)

```
<Bodys>
    <Cmd Name="MotionEnd"
Status="Send"/>
</Bodys>
```

Notes:
When using the motionbegin command, the robot needs to be in PLAY mode to start the movement.

Notes:
When using command 6 to send several commands, all Move command should stay between <Data> and be separated by \n.

June-20-2019 MotionBegin command

Example code:

```
action_str_13 = str('<Bodys> <Cmd Name="MotionBegin" Status="Send"> <Param
FileName = "sampletest.txt" StartLine = "0"/> <Data>MoveJ(150.000,0.000,0.000,-
56.000,60.000,56.000,0.000,0.000,0.000,0.000,0.000,0.000,0.000,0.000,0.000,0.000,0.
000,0.000,20,5,1,0,00,04)##MoveJ S=20% A=5% T=00 U=04</Data></Cmd>
</Bodys>')
```

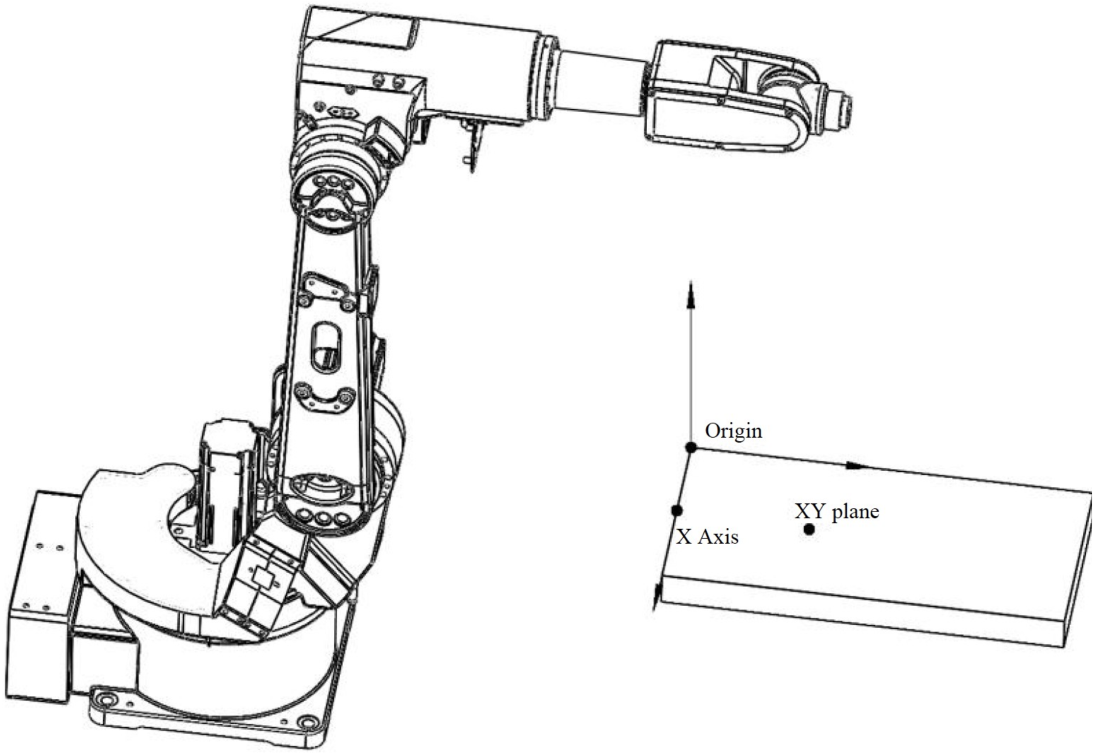
sampletest.txt
The name of files in the system which including following commands.

MoveJ(000)
First 9 numbers means User value: X,Y,Z,RX,RY,RZ,J7,J8,J9. Here we will use first 6 values.
Following 9 numbers means Joint value: J1~J9.They are useless here even you set them randomly.
Last 6 numbers means speed, acceleration, ?, ?, Tool number, User number.

Note:
Mutli-command should be separated by \n.

Note:
We need to set different Tool number and User number for different tools. The setting method can be found in the operation manual

June-20-2019 Setting User coordinate system



User coordinate system setting:
(3-point method)

1. Determine the required user coordinate system.

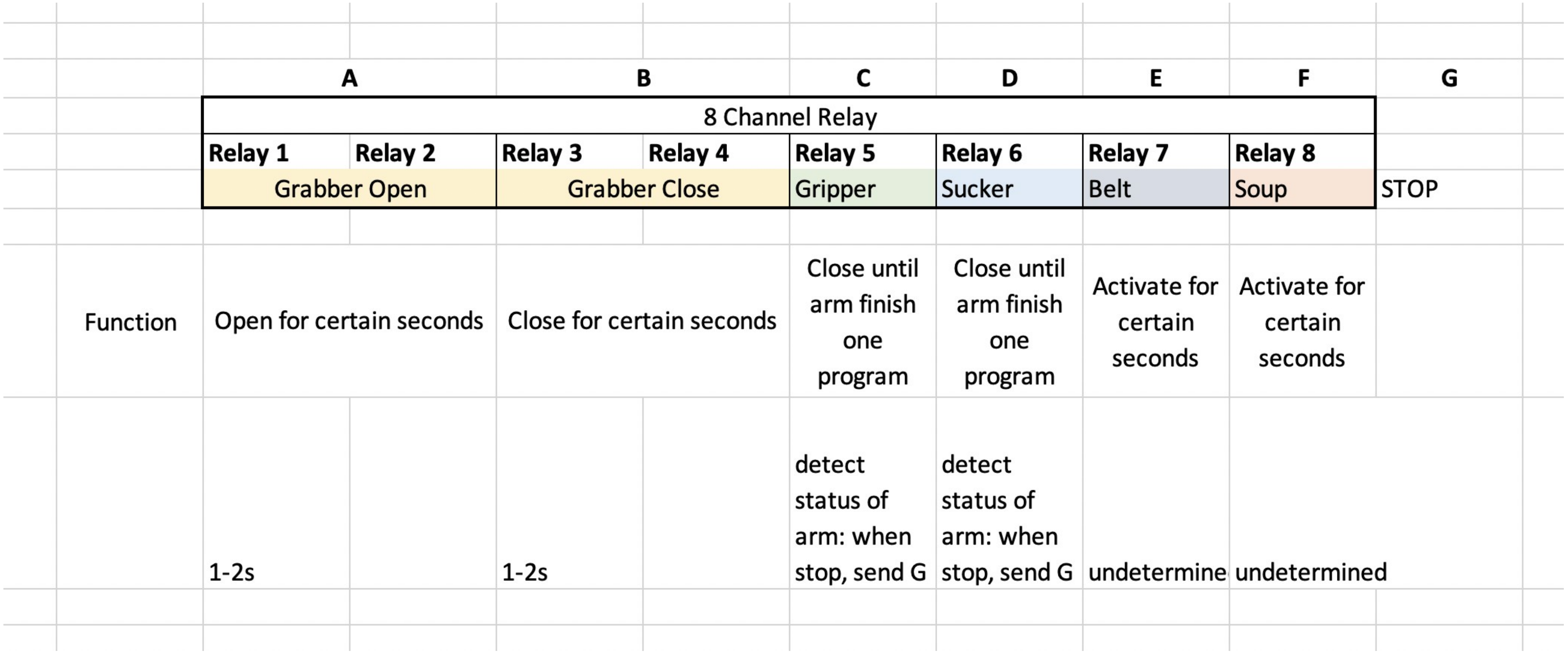
2. Move the robot with the operating tool to the origin of the coordinate system and record the point.

3. Move the robot with the operating tool to any point on the X-axis of the
4. coordinate system and record the point.

5. Move the robot with the operating tool to any point in the XY plane that belongs to the first quadrant and record the point.

6. Save the calculation results.

Aug-13-2019 Relay Connection



	A		B		C	D	E	F	G
	8 Channel Relay								
	Relay 1	Relay 2	Relay 3	Relay 4	Relay 5	Relay 6	Relay 7	Relay 8	
	Grabber Open		Grabber Close		Gripper	Sucker	Belt	Soup	STOP
Function	Open for certain seconds		Close for certain seconds		Close until arm finish one program	Close until arm finish one program	Activate for certain seconds	Activate for certain seconds	
	1-2s		1-2s		detect status of arm: when stop, send G	detect status of arm: when stop, send G	undetermine	undetermined	

Aug-13-2019 Connection with Raspberry

```
import socket

# Set the server address. i.e., the host address.
TCP_IP = '192.168.1.100'        #IP address should be computer ip address
TCP_PORT = 2222                 #Port should be same as port in the program in raspberry

# Set the buffer size to store the data.
BUFFER_SIZE = 20

# Create the socket
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

# Bind to the address
s.bind((TCP_IP, TCP_PORT))

# Wait for the incoming connection
s.listen(1)
print("Server Started")

# Accept the incoming connection if any.
conn, addr = s.accept()

# Print the IP Details
print ('Connection address:', addr)

# Start the loop.
while 1:
   MESSAGE = input()

   conn.send(MESSAGE.encode('utf-8'))

   # Start receiving the data.
   recvd_data = conn.recv(BUFFER_SIZE)

   # Print the received data.
   print("Command Received: ", recvd_data.decode('utf-8'))
# Close the connection once the program exits.
conn.close()
```

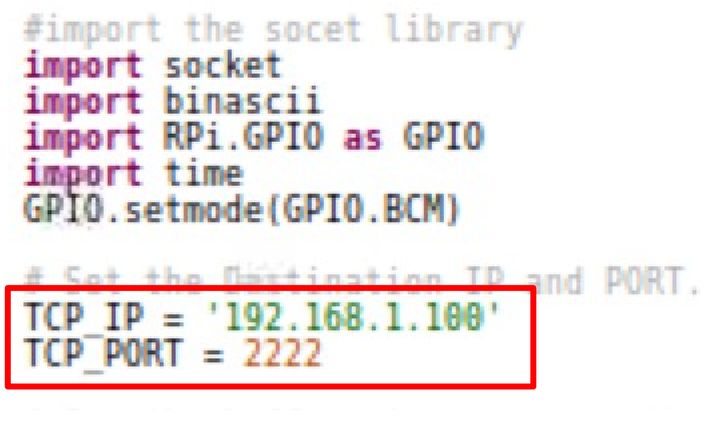
Aug-13-2019 Connection program in Raspberry

IP address:192.168.1.127

Location: /home/pi/Desktop/tcpip-testing/client_new.py

Check the ip address and port are as same as numbers in the ordering system

Run program on the computer first, after server has been established, run the program on the raspberry.



```
#import the socet library
import socket
import binascii
import RPi.GPIO as GPIO
import time
GPIO.setmode(GPIO.BCM)

# Set the Destination IP and PORT.
TCP_IP = '192.168.1.100'
TCP_PORT = 2222
```

Aug-13-2019 Relay & Arm Integration 1

```
action_str = str('<Bodys><Cmd Name="MotionEnd" Status="Send"/></Bodys>')
tn.write((action_str + "\n").encode('ascii'))
response = tn.read_until(b'/Bodys')
time.sleep(1)
action_str_2 = str('<Bodys> <Cmd Name="MotionBegin" Status="Send"> <Param FileName = "toolgrabwater1-1.txt" StartLine = "0"/> </Cmd> </Bodys>')
tn.write((action_str_2 + "\n").encode('ascii'))
print("\nWater Ordering.\n")
response = tn.read_until(b'/Bodys')

#detecting status of arm
time.sleep(1)
status = 'be'
while status != 'st':
  #  time.sleep(0.5)
  action_str_4 = str('<Bodys> <Cmd Name="GetMotionStatus" Status="Send"/> </Bodys>')
  tn.write((action_str_4 + "\n").encode('ascii'))
  response = tn.read_until(b'/Bodys>')
  print (response)
  status = response[86:88].decode("utf-8")

time.sleep(1)
MESSAGE = 'B'
conn.send(MESSAGE.encode('utf-8'))
# Start receiving the data.
recvd_data = conn.recv(BUFFER_SIZE)
# Print the received data.
print("Command Received: ", recvd_data.decode('utf-8'))

time.sleep(2)              ## should be as same as the time in raspberry

action_str = str('<Bodys><Cmd Name="MotionEnd" Status="Send"/></Bodys>')
tn.write((action_str + "\n").encode('ascii'))
response = tn.read_until(b'/Bodys')
time.sleep(1)
action_str_2 = str('<Bodys> <Cmd Name="MotionBegin" Status="Send"> <Param FileName = "toolgrabwater1-2.txt" StartLine = "0"/> </Cmd> </Bodys>')
tn.write((action_str_2 + "\n").encode('ascii'))
response = tn.read_until(b'/Bodys')
```

Aug-13-2019 Relay & Arm Integration 2

```
action_str = str('<Bodys><Cmd Name="MotionEnd" Status="Send"/></Bodys>')
tn.write((action_str + "\n").encode('ascii'))
response = tn.read_until(b'/Bodys')
time.sleep(1)
action_str_2 = str('<Bodys> <Cmd Name="MotionBegin" Status="Send"> <Param FileName = "toolsucbowl1-1.txt" StartLine = "0"/> </Cmd> </Bodys>')
tn.write((action_str_2 + "\n").encode('ascii'))
print("\nWater Ordering.\n")
response = tn.read_until(b'/Bodys')

#detecting status of arm
time.sleep(1)
status = 'be'
while status != 'st':
  #  time.sleep(0.5)
  action_str_4 = str('<Bodys> <Cmd Name="GetMotionStatus" Status="Send"/> </Bodys>')
  tn.write((action_str_4 + "\n").encode('ascii'))
  response = tn.read_until(b'/Bodys>')
  print (response)
  status = response[86:88].decode("utf-8")

time.sleep(1)
MESSAGE = 'D'
conn.send(MESSAGE.encode('utf-8'))
# Start receiving the data.
recvd_data = conn.recv(BUFFER_SIZE)
# Print the received data.
print("Command Received: ", recvd_data.decode('utf-8'))

time.sleep(2)

MESSAGE = 'G'                                  # C & D relay need to send G command from code
conn.send(MESSAGE.encode('utf-8'))
Start receiving the data.
recvd_data = conn.recv(BUFFER_SIZE)
Print the received data.
print("Command Received: ", recvd_data.decode('utf-8'))


action_str = str('<Bodys><Cmd Name="MotionEnd" Status="Send"/></Bodys>')
tn.write((action_str + "\n").encode('ascii'))
response = tn.read_until(b'/Bodys')
time.sleep(1)
action_str_2 = str('<Bodys> <Cmd Name="MotionBegin" Status="Send"> <Param FileName = "toolsucbowl1-2.txt" StartLine = "0"/> </Cmd> </Bodys>')
tn.write((action_str_2 + "\n").encode('ascii'))
response = tn.read_until(b'/Bodys')
```