

CPP to CPP Topic 통신(송신) HW2

```
1  #include "rclcpp/rclcpp.hpp"
2  #include "std_msgs/msg/string.hpp"
3  #include "std_msgs/msg/int32.hpp"
4  #include "std_msgs/msg/float64.hpp"
5  #include <chrono>
6  #include <memory>
7  #include <functional>
8
9
10 // class MyCppNode : public rclcpp::Node
11 // {
12 //     public:
13 //         MyCppNode() : Node("my_cpp_node")
14 //         {
15 //             RCLCPP_INFO(this->get_logger(), "Hello, ROS 2 C++ Node!");
16 //         }
17 // };
18
19 class MyNode : public rclcpp::Node
20 {
21     public:
22     MyNode();
23
24
25     private:
26     rclcpp::Publisher<std_msgs::msg::Float64>::SharedPtr publisher_F;
27     rclcpp::Publisher<std_msgs::msg::Int32>::SharedPtr publisher_I;
28     rclcpp::Publisher<std_msgs::msg::String>::SharedPtr publisher_S;
29     rclcpp::TimerBase::SharedPtr timer_;
30
31
32     int count_=0;
33
34     void timer_callback();
35
36
37 };
```

- ROS topic 통신을 위한 헤더파일

- Node 클래스 상속받는 MyNode 생성

- Rclcpp에서 실수, 정수, 문자열로 송신하는 변수들

- 시간 함수(전송 주기용)

- 송신 카운트 세는 변수

```
#include "../include/my_first_ros_rclcpp_pkg/publisher.hpp"

MyNode::MyNode() : Node("p")
{
    publisher_S = this->create_publisher<std_msgs::msg::String>("tS", 10);
    publisher_I = this->create_publisher<std_msgs::msg::Int32>("tI", 10);
    publisher_F = this->create_publisher<std_msgs::msg::Float64>("tF", 10);
    timer_ = this->create_wall_timer(std::chrono::duration<double>(1.0), std::bind(&MyNode::timer_callback, this));
}

void MyNode::timer_callback() {

    auto msgS = std_msgs::msg::String();
    msgS.data = "Hello World : " + std::to_string(count_++);
    RCLCPP_INFO(this->get_logger(), "Published message S: '%s'", msgS.data.c_str());
    publisher_S->publish(msgS);

    auto msgI = std_msgs::msg::Int32();
    msgI.data = count_;
    RCLCPP_INFO(this->get_logger(), "Published message I: '%d'", msgI.data);
    publisher_I->publish(msgI);

    auto msgF = std_msgs::msg::Float64();
    msgF.data = (count_/3.14);
    RCLCPP_INFO(this->get_logger(), "Published message F: '%f'", msgF.data);
    publisher_F->publish(msgF);

}

int main(int argc, char ** argv)
{
    rclcpp::init(argc, argv);
    auto node = std::make_shared<MyNode>();
    rclcpp::spin(node);
    rclcpp::shutdown();
    return 0;
}
```

CPP to CPP Topic 통신(송신) HW2

- 헤더파일 Node 클래스 기반 클래스 생성
- 각 변수마다 수신할 형태 세팅
- 타이머 변수: 특정 시간 지날때 마다 timer callback 함수 실행
- Callback함수
- msgS를 문자형 데이터로 설정, count값과 "Hello World" 저장
- 글자와 숫자를 문자로 변환하여 전송
- msgI를 정수형 데이터로 설정, count 값 저장
- 숫자를 송신함
- msgF를 실수형 데이터로 설정, count/3.14 값 저장
- 실수형으로 전송
- 계속하여 전송함 및 출력함

```

1  #include "rclcpp/rclcpp.hpp"
2  #include "std_msgs/msg/string.hpp"
3  #include "std_msgs/msg/int32.hpp"
4  #include "std_msgs/msg/float64.hpp"
5  #include <chrono>
6  #include <memory>
7  #include <functional>
8
9  class MyNode : public rclcpp::Node
10 {
11     public:
12     MyNode();
13
14     private:
15
16     rclcpp::Subscription<std_msgs::msg::String>::SharedPtr subS;
17     rclcpp::Subscription<std_msgs::msg::Int32>::SharedPtr subI;
18     rclcpp::Subscription<std_msgs::msg::Float64>::SharedPtr subF;
19
20     void topic_callbackS(const std_msgs::msg::String::SharedPtr msgS);
21     void topic_callbackI(const std_msgs::msg::Int32::SharedPtr msgI);
22     void topic_callbackF(const std_msgs::msg::Float64::SharedPtr msgF);
23
24     ~MyNode() {}

```

CPP to CPP Topic 통신(수신) HW2

- ROS topic 통신을 위한 헤더파일
- Node 클래스 상속받는 MyNode 생성
- Rclcpp에서 실수, 정수, 문자열로 수신하는 변수들
- 각 형에 맞춰 설정한 함수들 선언

CPP to CPP Topic 통신(송신) HW2

```
#include "../include/my_first_ros_rclcpp_pkg/subscriber.hpp"

MyNode::MyNode() : Node("s")
{
    subS=this->create_subscription<std_msgs::msg::String>("tS",0,std::bind(&MyNode::topic_callbackS,this,std::placeholders::_1));
    subI=this->create_subscription<std_msgs::msg::Int32>("tI",0,std::bind(&MyNode::topic_callbackI,this,std::placeholders::_1));
    subF=this->create_subscription<std_msgs::msg::Float64>("tF",0,std::bind(&MyNode::topic_callbackF,this,std::placeholders::_1));
}

void MyNode::topic_callbackS(const std_msgs::msg::String::SharedPtr msgS) {
    RCLCPP_INFO(this->get_logger(), "Published message S: '%s'",msgS->data.c_str());
}

void MyNode::topic_callbackI(const std_msgs::msg::Int32::SharedPtr msgI) {
    RCLCPP_INFO(this->get_logger(), "Published message I: '%d'",msgI->data);
}

void MyNode::topic_callbackF(const std_msgs::msg::Float64::SharedPtr msgF) {
    RCLCPP_INFO(this->get_logger(), "Published message F: '%f'",msgF->data);
}

int main(int argc, char ** argv)
{
    rclcpp::init(argc, argv);
    auto node = std::make_shared<MyNode>();
    rclcpp::spin(node);
    rclcpp::shutdown();
    return 0;
}
```

- 헤더파일 Node 클래스 기반 클래스 생성
- 각 변수의 대역폭에 맞게 입력 받은 데이터 변수에 저장
- 각 변수들마다 입력이 들어오면 해당 데이터를 형식에 따라 터미널에 출력
- 계속하여 수신 및 출력함

```

1 import rclpy
2 from rclpy.node import Node
3 from std_msgs.msg import String, Int32, Float64
4
5
6 class hwp(Node):
7     def __init__(self):
8         super().__init__('hp')
9         self.hpS=self.create_publisher(String, 'S',10)
10        self.hpI=self.create_publisher(Int32, 'I',10)
11        self.hpF=self.create_publisher(Float64, 'F',10)
12        self.timer=self.create_timer(1,self.phm)
13        self.count=0
14
15    def phm(self):
16        msgS=String()
17        msgS.data='Hello world:{0}'.format(self.count)
18        self.hpS.publish(msgS)
19        self.get_logger().info('Publisihed msg {0}'.format(msgS.data))
20
21
22        msgI=Int32()
23        msgI.data=self.count
24        self.hpI.publish(msgI)
25        self.get_logger().info('Publisihed msg {0}'.format(msgI.data))
26
27        msgF=Float64()
28        msgF.data=self.count/3
29        self.hpF.publish(msgF)
30        self.get_logger().info('Publisihed msg {0}'.format(msgF.data))
31
32        self.count+=1
33
34    def main(args=None):
35        rclpy.init(args=args)
36        node= hwp()
37        try:
38            rclpy.spin(node)
39        except KeyboardInterrupt:
40            node.get_logger().info('KeyboardInterrupt(SIGINT)')
41        finally:
42            node.destroy_node()
43            rclpy.shutdown()
44    if __name__ == '__main__':
45        main()

```

- 헤더파일 불러옴
- Class 설정
- CPP처럼 변수마다 통신할 대역폭(S,I,F) 및 형식 지정
- Timer와 count도 동일
- 문자열로 형태로 문장과 count 저장
- 터미널에 띄우기 및 전송 하는 명령어 설정
- 정수 형태로 문자열과 동일하게 설정
- 실수 형태로 문자열과 동일하게 설정
- Main 함수
- 각 데이터 전송 시도
- 키보드가 입력되면 특정 문장 출력
- 끝나면 기존에 있던 것들 다 삭제 및 종료
- 현재 창이 main창이면 main 함수 실행

```

1  import rclpy
2  from rclpy.node import Node
3  from std_msgs.msg import String, Int32, Float64
4
5
6  class hws(Node):
7      def __init__(self):
8          super().__init__('hs')
9          self.hsS=self.create_subscription(String,'S',self.stm,10)
10         self.hsI=self.create_subscription(Int32,'I',self.stm,10)
11         self.hsF=self.create_subscription(Float64,'F',self.stm,10)
12
13
14         def stm(self,msg):
15             self.get_logger().info('Receoved msg {0}'.format(msg.data))
16
17
18     def main(args=None):
19         rclpy.init(args=args)
20         node= hws()
21         try:
22             rclpy.spin(node)
23         except KeyboardInterrupt:
24             node.get_logger().info('Keyboard Interrupt (SIGINT)')
25         finally:
26             node.destroy_node()
27             rclpy.shutdown()
28
29     if __name__ == '__main__':
30         main()

```

- 헤더파일 불러옴

- Class 설정

- CPP처럼 변수마다 통신할 대역폭(S,I,F) 및 형식에 따라 데이터 개별 저장

- 터미널에 각 데이터 띄우는 함수

- Class에 있는 함수들 실행 시도

- 인터럽트 발생시 터미널에 특정 문장 출력

- 끝나면 모든 작업 삭제 및 중지

- Main 화면에서 실행시 main 함수 실행

CPP to Py Topic 통신(수신) HW2

```
1 #include "rclcpp/rclcpp.hpp"
2 #include "std_msgs/msg/string.hpp"
3 #include "std_msgs/msg/int32.hpp"
4 #include "std_msgs/msg/float64.hpp"
5 #include <chrono>
6 #include <memory>
7 #include <functional>
8
9 // class MyCppNode : public rclcpp::Node
10 // {
11 //     public:
12 //         MyCppNode() : Node("my_cpp_node")
13 //         {
14 //             RCLCPP_INFO(this->get_logger(), "Hello, ROS 2 C++ Node!");
15 //         }
16 //     };
17
18 class MyNode : public rclcpp::Node
19 {
20     public:
21         MyNode();
22
23     private:
24         rclcpp::Publisher<std_msgs::msg::Float64>::SharedPtr publisher_F;
25         rclcpp::Publisher<std_msgs::msg::Int32>::SharedPtr publisher_I;
26         rclcpp::Publisher<std_msgs::msg::String>::SharedPtr publisher_S;
27         rclcpp::TimerBase::SharedPtr timer_;
28
29         int count_=0;
30
31         void timer_callback();
32 };
33
34 #include "../include/my_first_ros_rclcpp_pkg/publisher.hpp"
35
36 MyNode::MyNode() : Node("p")
37 {
38     publisher_S = this->create_publisher<std_msgs::msg::String>("S", 10);
39     publisher_I = this->create_publisher<std_msgs::msg::Int32>("I", 10);
40     publisher_F = this->create_publisher<std_msgs::msg::Float64>("F", 10);
41     timer_ = this->create_wall_timer(std::chrono::duration<double>(1.0), std::bind(&MyNode::timer_callback, this));
42 }
43
44 void MyNode::timer_callback() {
45     auto msgS = std_msgs::msg::String();
46     msgS.data = "Hello World : " + std::to_string(count_++);
47     RCLCPP_INFO(this->get_logger(), "Published message S: '%s'", msgS.data.c_str());
48     publisher_S->publish(msgS);
49
50     auto msgI = std_msgs::msg::Int32();
51     msgI.data = count_;
52     RCLCPP_INFO(this->get_logger(), "Published message I: '%d'", msgI.data);
53     publisher_I->publish(msgI);
54
55     auto msgF = std_msgs::msg::Float64();
56     msgF.data = (count_ / 3.14);
57     RCLCPP_INFO(this->get_logger(), "Published message F: '%f'", msgF.data);
58     publisher_F->publish(msgF);
59 }
60
61 int main(int argc, char ** argv)
62 {
63     rclcpp::init(argc, argv);
64     auto node = std::make_shared<MyNode>();
65     rclcpp::spin(node);
66     rclcpp::shutdown();
67     return 0;
68 }
```

- 이전 CPP 수신 파일과 동일
- 단 각 변수별 대역폭이(S, I, F) Py 송신부와 일치하도록 변경

CPP to Py Topic 통신(송신) HW2

- 이전 Py 송신 파일과 동일

- CPP 수신 노드의 형식 별 대역폭과 Py 송신 노드의 형식 별 대역폭이 동일

```
1 import rclpy
2 from rclpy.node import Node
3 from std_msgs.msg import String, Int32, Float64
4
5
6 class hws(Node):
7     def __init__(self):
8         super().__init__('hs')
9         self.hsS=self.create_subscription(String,'S',self.stm,10)
10        self.hsI=self.create_subscription(Int32,'I',self.stm,10)
11        self.hsF=self.create_subscription(Float64,'F',self.stm,10)
12
13
14    def stm(self,msg):
15        self.get_logger().info('Receoved msg {0}'.format(msg.data))
16
17
18 def main(args=None):
19     rclpy.init(args=args)
20     node= hws()
21     try:
22         rclpy.spin(node)
23     except KeyboardInterrupt:
24         node.get_logger().info('Keyboard Interrupt (SIGINT)')
25     finally:
26         node.destroy_node()
27         rclpy.shutdown()
28 if __name__=='__main__':
29     main()
```



```
src > my_first_ros_rclcpp_pkg > src > publisher.cpp > @ timer_callback()
1 #include "../include/my_first_ros_rclcpp_pkg/publisher.hpp"
2
3 MyNode::MyNode() : Node("p")
4 {
5     publisher_S = this->create_publisher<std_msgs::msg::String>("S", 10);
6     publisher_I = this->create_publisher<std_msgs::msg::Int32>("I", 10);
7     publisher_F = this->create_publisher<std_msgs::msg::Float64>("F", 10);
8     timer_ = this->create_wall_timer(std::chrono::duration<double>(1.0), std::bind(&MyNode::timer_callback, this));
9 }
10
11 void MyNode::timer_callback() {
12
13     auto msgS = std_msgs::msg::String();
14     msgS.data = "Hello World" + std::to_string(count++) + " ";
15     RCLCPP_INFO(this->get_logger(), "Published message S: '%s'", msgS.data.c_str());
16     publisher_S->publish(msgS);
17
18     auto msgI = std_msgs::msg::Int32();
19     msgI.data = count;
20     RCLCPP_INFO(this->get_logger(), "Published message I: '%d'", msgI.data);
21     publisher_I->publish(msgI);
22
23     auto msgF = std_msgs::msg::Float64();
24     msgF.data = (count/2.14);
25     RCLCPP_INFO(this->get_logger(), "Published message F: '%f'", msgF.data);
26     publisher_F->publish(msgF);
27 }
28
29 int main(int argc, char ** argv)
30 {
31     rclcpp::init(argc, argv);
32     auto node = std::make_shared<MyNode>();
33     rclcpp::spin(node);
34     rclcpp::shutdown();
35     return 0;
36 }
```

```
1 sub.py > @ main
2 import rclpy
3 from rclpy.node import Node
4 from std_msgs.msg import String, Int32, Float64
5
6 class hws(Node):
7     def __init__(self):
8         super().__init__('hs')
9         self.hsS=self.create_subscription(String, 'S', self.stm, 10)
10        self.hsI=self.create_subscription(Int32, 'I', self.stm, 10)
11        self.hsF=self.create_subscription(Float64, 'F', self.stm, 10)
12
13    def stm(self, msg):
14        self.get_logger().info('Received msg {0}'.format(msg.data))
15
16 def main(args=None):
17     rclpy.init(args=args)
18     node= hws()
19     try:
20         rclpy.spin(node)
21     except KeyboardInterrupt:
22         node.get_logger().info('Keyboard Interrupt (SIGINT)')
23     finally:
24         node.destroy_node()
25         rclpy.shutdown()
26 if __name__ == '__main__':
27     main()
```

```
[INFO] [1757665014.058053279] [p]: Published message I: '19'
[INFO] [1757665014.058066515] [p]: Published message S: '6.050955'
[INFO] [1757665015.057910218] [p]: Published message S: 'Hello World : 19'
[INFO] [1757665015.058178563] [p]: Published message I: '20'
[INFO] [1757665015.058217929] [p]: Published message F: '6.369427'
[INFO] [1757665016.057989307] [p]: Published message S: 'Hello World : 20'
[INFO] [1757665016.058338881] [p]: Published message I: '21'
[INFO] [1757665016.058377080] [p]: Published message F: '6.687898'
[INFO] [1757665017.058019565] [p]: Published message S: 'Hello World : 21'
[INFO] [1757665017.058190883] [p]: Published message I: '22'
[INFO] [1757665017.058224077] [p]: Published message F: '7.006369'
[INFO] [1757665018.058060590] [p]: Published message S: 'Hello World : 22'
[INFO] [1757665018.058311200] [p]: Published message I: '23'
[INFO] [1757665018.058341260] [p]: Published message F: '7.324841'
^C[INFO] [1757665018.848121050] [rclcpp]: signal_handler(signum=2)
davidnomal:~/intern_ws/R052/test/colcon_ws$
```

```
[INFO] [1757665011.058551187] [hs]: Received msg Hello World : 15
[INFO] [1757665011.058840081] [hs]: Received msg 16
[INFO] [1757665011.059184692] [hs]: Received msg 5.095541401273885
[INFO] [1757665012.058720504] [hs]: Received msg Hello World : 16
[INFO] [1757665012.059330444] [hs]: Received msg 17
[INFO] [1757665012.060293982] [hs]: Received msg 5.414012738853503
[INFO] [1757665012.059434130] [hs]: Received msg Hello World : 17
[INFO] [1757665013.060247280] [hs]: Received msg 18
[INFO] [1757665013.061258938] [hs]: Received msg 5.732484076433121
[INFO] [1757665014.059432154] [hs]: Received msg Hello World : 18
[INFO] [1757665014.060049999] [hs]: Received msg 19
[INFO] [1757665014.060609893] [hs]: Received msg 6.050955414012739
[INFO] [1757665015.059940632] [hs]: Received msg Hello World : 19
[INFO] [1757665015.060978092] [hs]: Received msg 20
[INFO] [1757665015.061893117] [hs]: Received msg 6.369426751592356
[INFO] [1757665016.059880874] [hs]: Received msg Hello World : 20
[INFO] [1757665016.060653396] [hs]: Received msg 21
[INFO] [1757665016.061461095] [hs]: Received msg 6.687898089171974
[INFO] [1757665017.059778107] [hs]: Received msg Hello World : 21
[INFO] [1757665017.060522635] [hs]: Received msg 22
[INFO] [1757665017.061160217] [hs]: Received msg 7.006369426751592
[INFO] [1757665018.059566097] [hs]: Received msg Hello World : 22
[INFO] [1757665018.059928091] [hs]: Received msg 23
[INFO] [1757665018.060156739] [hs]: Received msg 7.32484076433121
```

CPP to Py Topic 통신