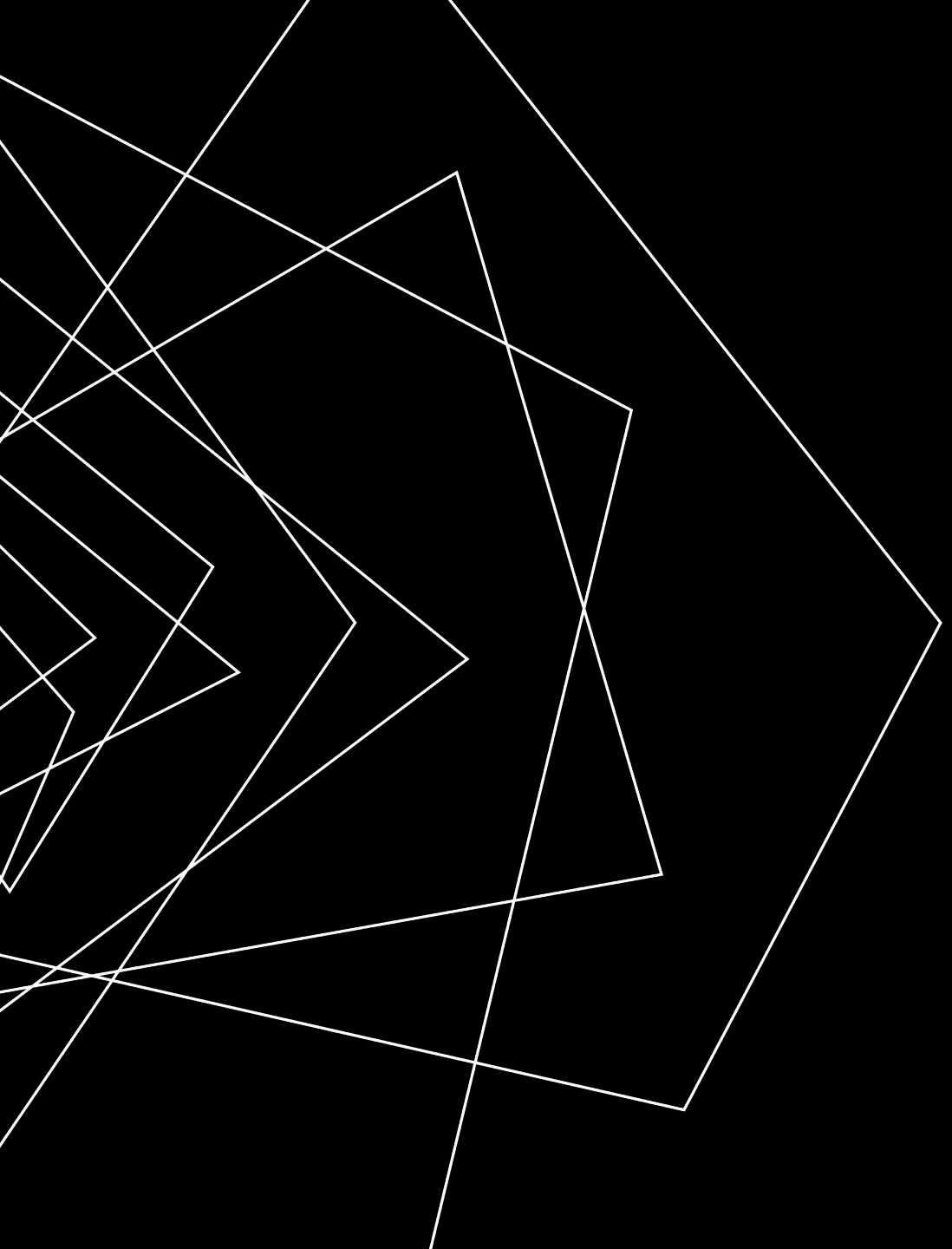


ROS2 TEMPERATURE MONITOR

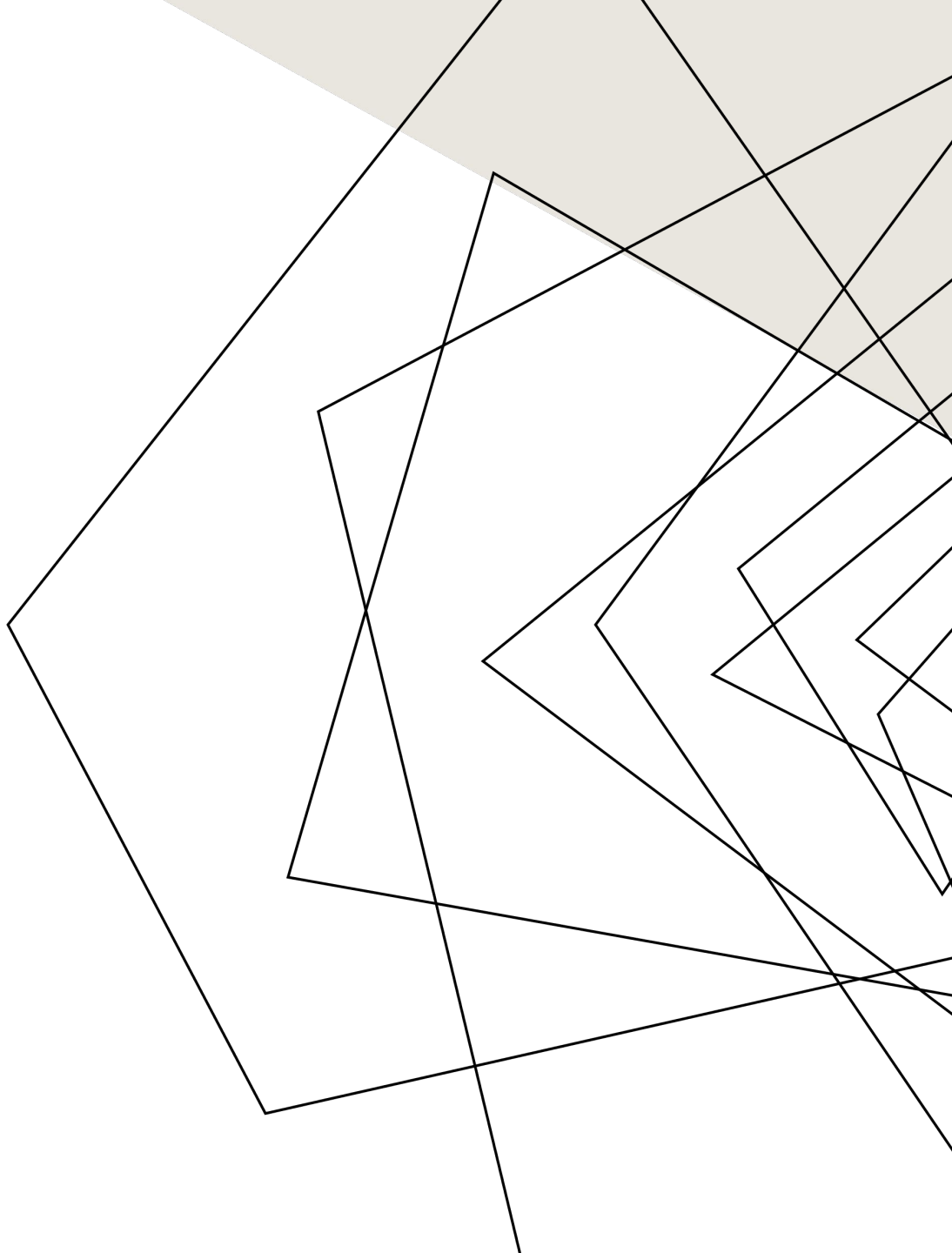
Presented to:
Robotics-software-community

INTRODUCTION

- I am Youhana Beshay, a fresh ECE graduate from Ain shams university
- My relevant experience in the field :
    - GP: Developed an AMR using ROS 2 with an embodied agent

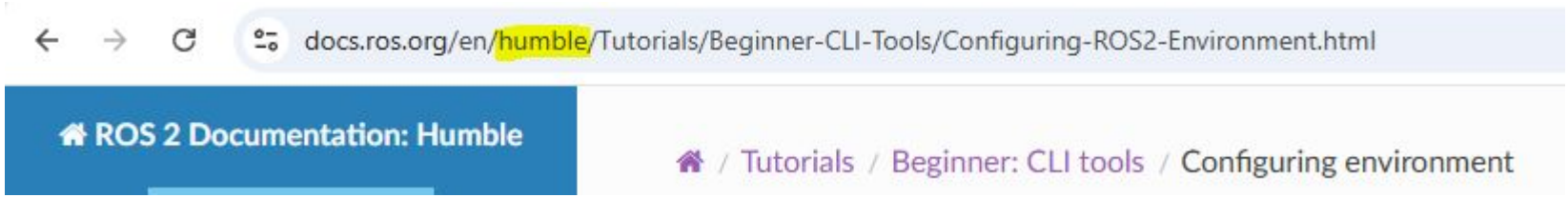
OVERVIEW

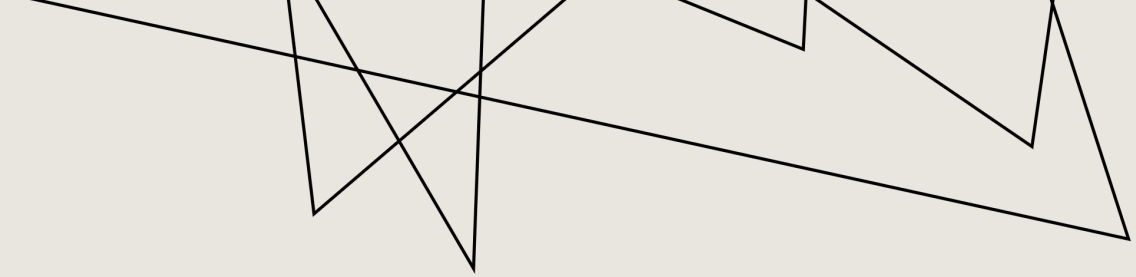
- This is not a comprehensive step-by-step guide
- Will first discus my approach and provide relevant resources
- The focus will be on challenges I faced and how I tackled them
- Feel free to stop me and ask any questions!



EXERCISE PREREQUISITES

- The exercise expects prior knowledge in basic Ros2 concepts, so we won't get into any details about them
    - But I can highly suggest relying on the Ros2 documentation:





TASK 1: ENVIRONMENT SETUP

FILE STRUCTURE

- We are required to make 3 packeges
  1. Custom message type (temeprature_interfaces)
  2. Python Publisher (temperature_pub)
  3. C++ Subscriber (temperature_sub)

```
1-temperature-monitor/
├──  packages/
```

FILE STRUCTURE

- We are required to make 3 packeges
  1. Custom message type (temeprature_interfaces)
  2. Python Publisher (temperature_pub)
  3. C++ Subscriber (temperature_sub)
- So we use ros2 pkg create for all 3

```
1-temperature-monitor/
├──  packages/
│    └── my_package/
│        ├── temeprature_interfaces/
│        ├── temperature_pub/
│        ├── temperature_sub/
```

FILE STRUCTURE

- We are required to make 3 packeges
  1. Custom message type (temeprature_interfaces)
  2. Python Publisher (temperature_pub)
  3. C++ Subscriber (temperature_sub)
- So we use ros2 pkg create for all 3
- We need the nodes to be containerized so we will need dockerfiles and docker-compose.yaml

```
1-temperature-monitor/
├──  packages/
│    └── my_package/
│        ├── temeprature_interfaces/
│        ├── temperature_pub/
│        │    └── Dockerfile
│        ├── temperature_sub/
│        │    └── Dockerfile
├── docker-compose.yaml
```

DOCKER FILES

- Useful Resources for dockerfile:
  - Articulated robotics Docker playlist.
  - Choosing Between RUN, CMD, and ENTRYPOINT blogpost.

- Useful Resources for docker-compose:
  - Configuring Ros2 environment documentation.
  - RoboticsUnveiled (ROS2 IPC, DDS..) article.

DOCKER FILE

```
FROM ros:humble

COPY temperature_interfaces docker_ws/src/temperature_interfaces
COPY temperature_pub docker_ws/src/temperature_pub


WORKDIR /docker_ws/

RUN . /opt/ros/humble/setup.sh && \
colcon build


CMD ["/bin/bash", "-c", "source /opt/ros/humble/setup.bash && \
source install/setup.bash &&\
ros2 run temperature_pub temp_pub" ]
```

DOCKER FILE

```
...

ARG USERNAME=ros
ARG USER_UID=1000
ARG USER_GID=$USER_UID

RUN groupadd --gid $USER_GID $USERNAME \
&& useradd -s /bin/bash --uid $USER_UID --gid $USER_GID -m $USERNAME \
&& mkdir /home/$USERNAME/.config && chown $USER_UID:$USER_GID /home/$USERNAME/.config


USER ros

CMD ["/bin/bash", "-c", "source /opt/ros/humble/setup.bash && \
source install/setup.bash &&\
ros2 run temperature_pub temp_pub" ]
```

DOCKER COMPOSE

```
services:

temperature_pub:

build:
  context: ./packages
  dockerfile: temperature_pub/Dockerfile

environment:
# Note ROS_LOCALHOST_ONLY should be changed to 0 if we change the network to bridge
- ROS_LOCALHOST_ONLY=1
# For Linux : (IDs 0-101 and 215-232)
- ROS_DOMAIN_ID=42

network_mode: host

container_name: temperature_publisher
```

DOCKER COMPOSE

```
services:

temperature_pub:

build:
  context: ./packages
  dockerfile: temperature_pub/Dockerfile

environment:
# Note ROS_LOCALHOST_ONLY should be changed to 0 if we change the network to bridge
- ROS_LOCALHOST_ONLY=1
# For Linux : (IDs 0-101 and 215-232)
- ROS_DOMAIN_ID=42

network_mode: host

ipc: host

container_name: temperature_publisher
```

INTER PROCESS COMMUNICATION (IPC)

- Serves as the underlying mechanism for communication between processes running on the **same system**.

- DDS uses IPC to allow for:
  1. **Efficient Local Communication:** by using shared memory.
  2. **Resource Sharing:** such as data buffers.
  3. **Low-latency Communication:** by bypassing network protocols.

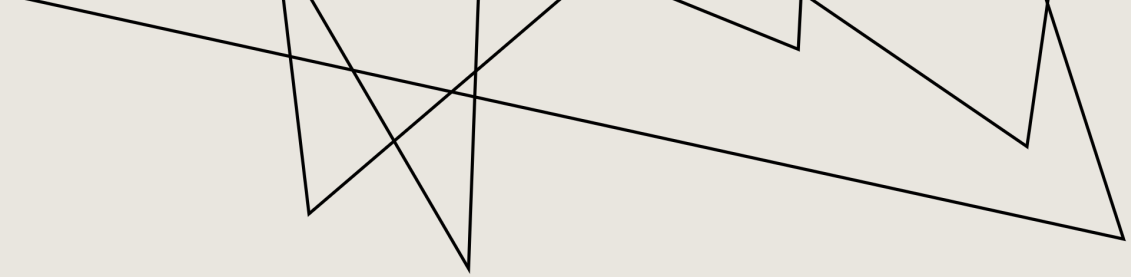
IPC is not needed/used in bridge network setup

TASK 2: MESSAGE DEFINITION

TEMPERATURE.MSG

```
float64 value
builtin_interfaces/Time time_stamp
string sensor_id
#Displaying unit should be handled inside publisher - defualt value = true
bool is_celsius True
```

- **And add relevant dependencies in the cmakelists and package.xml**



TASK 3: PYTHON PUBLISHER

PYTHON PUBLISHER

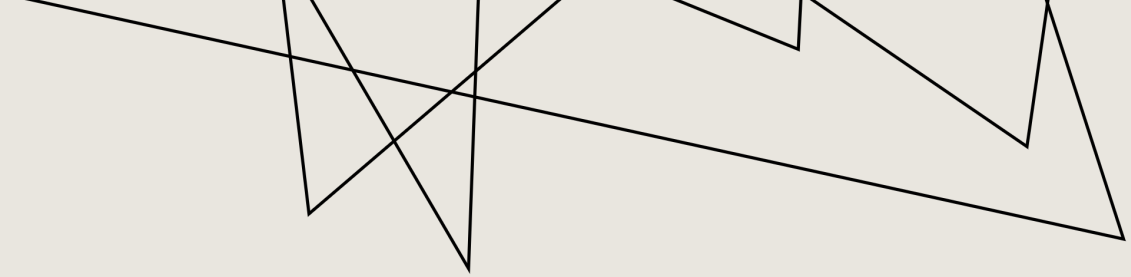
```python
class TemperaturePublisherNode(Node):
  def __init__(self):
    super().__init__("temperature_pub")

    # Parameter declaration and default values
    self.declare_parameter("publish_frequency", 1.0)
    self.declare_parameter("sensor_id", "sensor_1")
    self.declare_parameter("min_temp", 10)
    self.declare_parameter("max_temp", 50)

    # Assigning parameters to internal variables
    self.publish_frequency_ = self.get_parameter("publish_frequency").value
    self.sensor_id_ = self.get_parameter("sensor_id").value
    self.min_temp_ = self.get_parameter("min_temp").value
    self.max_temp_ = self.get_parameter("max_temp").value
```

PYTHON PUBLISHER

```python
self.temp_publisher_ = self.create_publisher(Temperature, "temperature")
self.timer_ = self.create_timer(1.0 / self.publish_frequency_, self.publish_temperature)
self.get_logger().info("Temperature publisher has been started.")

def publish_temperature(self):
  msg = Temperature()

  msg.value = self.randomize_temperature()

  msg.time_stamp = self.get_clock().now().to_msg()
  msg.sensor_id = self.sensor_id_
  # For clarity only as it's true by default
  msg.is_celsius = True

  self.get_logger().info(f'Published: {msg.value:.2f}°{"C" if msg.is_celsius else "F"} from {msg.sensor_id}')
  self.temp_publisher_.publish(msg)
```

PYTHON PUBLISHER

```python
def randomize_temperature(self):

  # Used clock for generation to simulate that values depend on time of "measurement"
  seconds=self.get_clock().now().nanoseconds / 1e9

  cycle_period_sec =200
  sin_phase_fraction= (seconds % cycle_period_sec)/cycle_period_sec

  # Generate a sine wave (it outputs [-1,1])
  sin_value=np.sin(2*np.pi*sin_phase_fraction)

  # Map [-1,1] to [min_temp,max_temp]
  temperature=(sin_value+1)/2 * (self.max_temp_-self.min_temp_) + self.min_temp_

  temperature_noisy=temperature + np.random.normal(0,1)

  return temperature_noisy
```



TASK 4:
C++
SUBSCRIBER

C++ SUBSCRIBER

```
public:
TemperatureSubscriberNode() : Node("temperature_sub")
{

// Parameter declaration and default values
this->declare_parameter("moving_average_period",10);
```

- **This parameter is used as the denominator in calculation**
- **So how should we avoid division by zero/-ve number ?**

C++ SUBSCRIBER

```
public:
TemperatureSubscriberNode() : Node("temperature_sub")
{

// moving_average_period must be > 0
auto mv_av_per_param_descriptor = rcl_interfaces::msg::ParameterDescriptor{};
mv_av_per_param_descriptor.integer_range = {rcl_interfaces::msg::IntegerRange()
                                                    .set__from_value(1)
                                                    .set__to_value(std::numeric_limits<int>::max())
                                                    .set__step(1)};


// Parameter declaration and default values
this->declare_parameter("moving_average_period",10,mv_av_per_param_descriptor);
```

C++ SUBSCRIBER

```cpp
void temperature_processing(const temperature_interfaces::msg::Temperature::SharedPtr msg)
{
double temp_celsius = msg->value;
// Convert to celsius if needed
if (!msg->is_celsius)
{
temp_celsius = (temp_celsius - 32) * 5 / 9;
}

// Update max temperature
if (temp_celsius > max_temp_)
{
max_temp_ = temp_celsius;
}

temp_celsius_values_.push(temp_celsius);
```

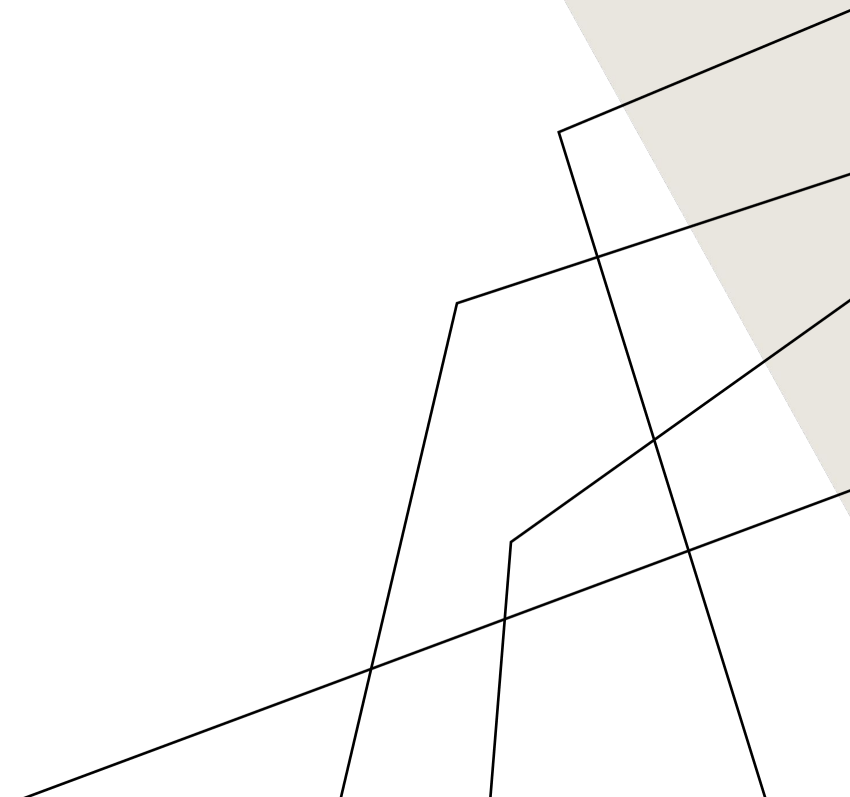
C++ SUBSCRIBER

```cpp
// Calculate moving average
sum_ += temp_celsius;
double mv_average = 0.00;
bool mv_average_ready = false;


// Cast moving average period to size_t to avoid warning at build
if (temp_celsius_values_.size() >= static_cast<size_t>(moving_average_period_))
{
    sum_ = sum_ - temp_celsius_values_.front();
    temp_celsius_values_.pop();
    mv_average = sum_ / moving_average_period_;

    // Flag for when to begin printing moving average values
    mv_average_ready = true;

    // Add to queue to be used in Trend calculation
    last_averages_.push(mv_average);
}
```

TASK 5: ADVANCED FEATURES

QOS SETTINGS

- Useful Resources :
  - QOS setting ROS 2 documentation.
  - QOS profiles header file github.



QOS SETTINGS

```
from rclpy.qos import qos_profile_sensor_data



class TemperaturePublisherNode(Node):
     def __init__(self):
         super().__init__("temperature_pub")

         self.temp_publisher_ = self.create_publisher(Temperature, "temperature", qos_profile_sensor_data)
```

GRACEFUL SHUTDOWN HANDLING

- Useful Resources :
  - Medium's How to use tini and init system in docker containers article
  - Fosslinux's "The ABCs of linux signals" article
  - Signal propagation by ros2 run verb github issue
  - How to gracefully shutdown a python node github issue

PYTHON PUBLISHER (GRACEFUL SHUTDOWN)

```python
def main(args=None):
  rclpy.init(args=args,signal_handler_options=SignalHandlerOptions.ALL)
  node = TemperaturePublisherNode()

  # Graceful shutdown handling
  try:
      rclpy.spin(node)
  except KeyboardInterrupt:
      print("KeyboardInterrupt caught: shutting down")
  except ExternalShutdownException:
      print("ExternalShutdownException caught: shutting down")
  except Exception as e:
      print(f"Exception caught: {str(e)}")
      print(traceback.format_exc())
  finally:
      node.destroy_node()
      if rclpy.ok():
                rclpy.shutdown()
      print("Publisher shutdown complete")
```

GRACEFUL SHUTDOWN SCENARIOS

1. Interrupting the natively run publisher :

```
youhana@youhana-virtual-machine:~$ ros2 run temperature_pub temp_pub
[INFO] [1756430803.368352723] [temperature_pub]: Temperature publisher has been started.
[INFO] [1756430804.295475298] [temperature_pub]: Published: 31.67°C from sensor_1
[INFO] [1756430805.294314487] [temperature_pub]: Published: 33.16°C from sensor_1
^CKeyboardInterrupt caught - shutting down
Publisher shutdown complete
```

   - Everything works as expected.✔️

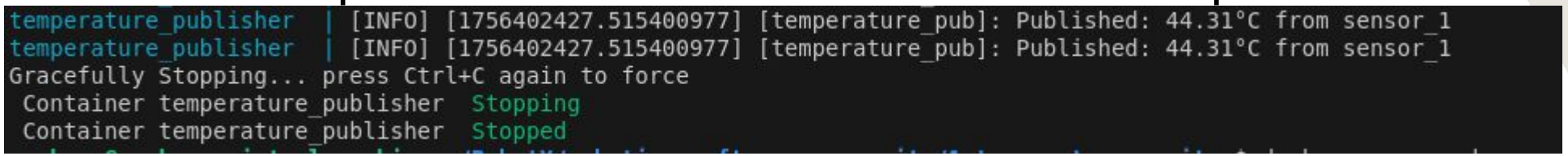
2. Interrupting the containerized publisher :

```
 Container temperature_publisher  Stopping
temperature_subscriber exited with code 0
temperature_publisher      | [INFO] [1756429224.219826341] [temperature_pub]: Published: 43.99°C from sensor_1
temperature_publisher      | [INFO] [1756429225.219797653] [temperature_pub]: Published: 44.49°C from sensor_1
temperature_publisher      | [INFO] [1756429226.219762930] [temperature_pub]: Published: 45.00°C from sensor_1
temperature_publisher      | [INFO] [1756429227.219706080] [temperature_pub]: Published: 44.59°C from sensor_1
temperature_publisher      | [INFO] [1756429228.219681313] [temperature_pub]: Published: 44.76°C from sensor_1
temperature_publisher      | [INFO] [1756429229.220252519] [temperature_pub]: Published: 44.73°C from sensor_1
temperature_publisher      | [INFO] [1756429230.220154540] [temperature_pub]: Published: 45.64°C from sensor_1
temperature_publisher      | [INFO] [1756429231.220086518] [temperature_pub]: Published: 47.55°C from sensor_1
temperature_publisher      | [INFO] [1756429232.220057378] [temperature_pub]: Published: 48.56°C from sensor_1
temperature_publisher      | [INFO] [1756429233.220028168] [temperature_pub]: Published: 47.40°C from sensor_1
 Container temperature_publisher  Stopped
```

   - SIGTERM Didn’t reach the publisher
   - Docker waits for 10 seconds then Sends SIGKILL❌

GRACEFUL SHUTDOWN SCENARIOS

3. Containerized publisher with **init: true** in docker compose



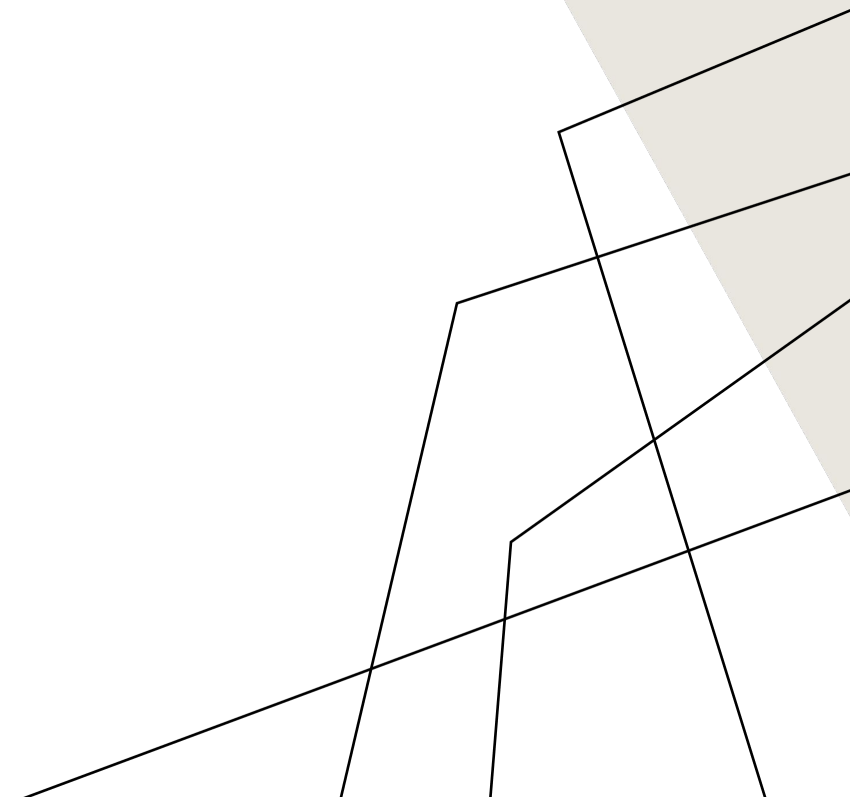
- Docker should send SIGTERM but it seems it sends SIGKILL Directly ❌

4. Containerized publisher running the publisher directly instead of using "ros2 run"

- Everything works as expected ✔️

LAUNCH FILE

- Useful Resources :
  - Incorrect signal handling in ros2 launch github issue



LAUNCH FILE

```python
from launch import LaunchDescription
from launch.actions import ExecuteProcess, LogInfo

# This launch file needs to be run in non interactive mode '-n' also stop_signal = SIGINT in docker compose
# As without this workaround containers will not close cleanly
# see https://github.com/ros2/launch/issues/666 for more info

# TODO: However this workaround still does not display nodes shutting down messages for some reason

def generate_launch_description():
  return LaunchDescription([
        LogInfo(msg='Starting Temperature Monitor containers.'),
        ExecuteProcess(
              cmd=['bash', '-c', 'docker compose -f docker-compose.yaml up --build'],
              name='Temp-Monitor',
              output='both',
              emulate_tty=True,
),

])
```

BONUS TASKS: CHALLENGES

ROS2 BRIDGE & WEB INTERFACE

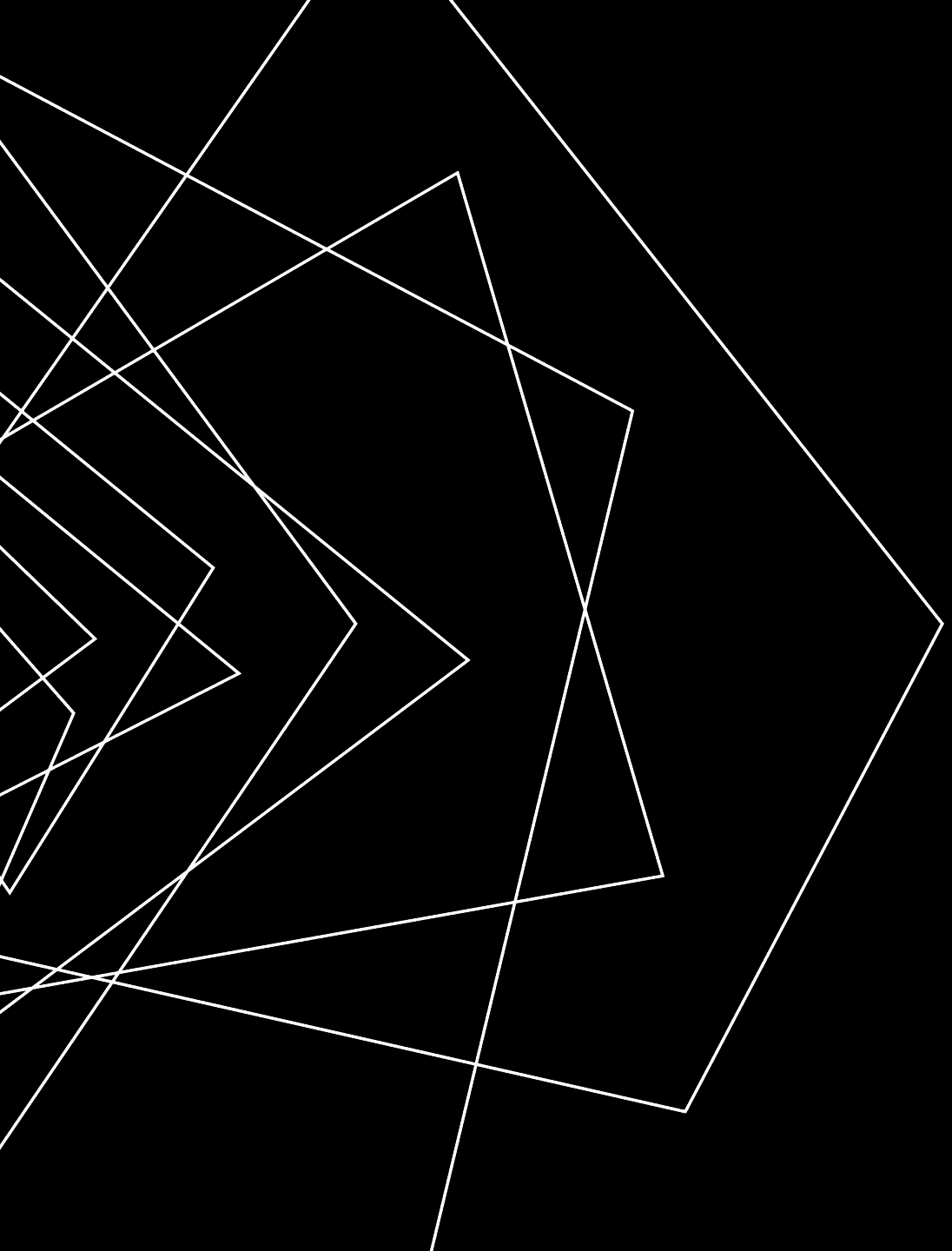
- Useful Resources :
  - FoxGlove's How to use Rosbridge in ROS2 article
  - Rosbridge_suite github repo
  - FreeCodeCamp’s “a beginner’s guide to websockets” video

DOCKER FILE

```
FROM ros:humble

COPY temperature_interfaces docker_ws/src/temperature_interfaces
COPY temperature_web docker_ws/temperature_web
# Install rosbridge-server
RUN apt-get update && apt-get install -y \
ros-humble-rosbridge-server\
&& rm -rf /var/lib/apt/lists/*

WORKDIR /docker_ws

RUN . /opt/ros/humble/setup.sh && \
colcon build

CMD ["/bin/bash", "-c", "source /opt/ros/humble/setup.bash && \
source install/setup.bash && \
echo 'Open your browser at: http://localhost:8080' && \
ros2 launch rosbridge_server rosbridge_websocket_launch.xml 'default_call_service_timeout:=5.0'
'call_services_in_new_thread:=true' 'send_action_goals_in_new_thread:=true' & \
python3 -m http.server 8080 --directory temperature_web "]
```

WEB INTERFACE

- **Rest of the code will be much easier to present in VS code directly with a live demo.*(if there's time)***



THANK YOU