# PROGRESS REVIEW 10


## JIMIT GANDHI
## ILR 9
## TEAM G Robographers

## Other Team members

**Rohit Dashrathi**
**Gauri Gandhi**
**Sida Wang**
**Tiffany May**

## Introduction

In this Progress Review these were the following goals:

1. New pan tilt design integration set up by Rohit.
2. Multi-master setup by Gauri and Jimit.
3. A program that decides which camera captures best expression by Jimit.
4. Single person detector in a multi-person environment by Sida and Jimit.
5. Pan tilt algorithm re-design by Tiffany.
6. Learning about flocking algorithm by Tiffany and Jimit.

## Individual Progress

Among the above goals, my major goals this time was to create a node that could decide which camera or robot gets the best smile expression and then request that robot to click the picture. In the other goals where I have mentioned two people (one being me and other team member), my contribution was basically helping the other team member in basically design and debugging.

So before I dive into the major goal, first thing me and Gauri tackled was the biggest challenge we had faced in our entire project. The correct configuration and package implementations of ROCON framework which I have explained in detail in my previous ILR. After we understood how ROCON works, I went through some tutorials of ROCON on ros website where they have explained how you can integrate your robot or customize ROCON package to connect your own robot. The people who have created ROCON package on ROS Indigo version, have actually introduced an abstract layered package which encapsulates the entire lower level working of ROCON. All one has to do is independently introduce your own package which contains robot functionalities into this abstract interface in the form of what they call as Rapps which stands for ROCON apps. Rapps are just packages which are taken as arguments in the ROCONs abstract package called as ROCON Capabilities package. After we create RAPPS, the Capabilities take the launch files of the Rapps and launch them in a file called concert-client.launch.

The Rapps basically is a package of four files. They are as follows

1. Launch file
2. Interface file
3. Rapps file
4. Parameters file

The launch file is just like any other launch file where we launch all the nodes we desire and also we mention the remappings of the topics as they will be different in the HUB namespace (HUB - a master which regulates the communication of all the other masters and also is responsible to launch them).
An interface file is where you define the topics the master or the nodes within that master would subscribe or publish. This file is of YAML format as it is used during compilation period which is

when you do Catkin_make. We can also enter services if we are using client/server response method of two way communication.
A parameter file is the one in which we can import parameters to the Parameter server that is local to one particular master in a multi-master setup. Here we can initiate the variables with different values for example the robot name or node name or port number etc.

A Rapp file is where we mention the names of the three other files along with their paths relative to this file on the machine. This file is actually taken as an argument in the Capabilities package launch file called concert_client.launch. So care has to be taken to make this file in the format which the ROCON creators have explained on their websites as well as their public Git repository also found on ros website. With this information I created a Rapp for the best photo selector which I will discuss in the following paragraph while Gauri created Rapp for the three Turtlebot clients.

Initially we tried to launch a simple number sending node on the HUB and number receiving nodes on other Turtlebot clients. Once this was successful, my next task was to replace this number sending node to the a node that subscribes to the emotion_talker node that Gauri created in the Turtlebot client Rapps, compare which turtlebot gets the best emotion output and then publish the id of that Turtlebot to all the Turtlebots. Once I subscribed to the three Turtlebots I realised that these emotion values were not in synchronization. That is the emotions received by my node at a particular time were emotion values of different times, approximately few millisecond difference and worst case even a second. So to make sure that we get synchronous readings, I introduced a Timestamp message type along with the emotion values.To do this I had to edit the header file and .msg files of the Intraface package. Once this was done, I created an intermediate node that would take in the emotion values from different masters and combine them if and only if they are of the same timestamp. If a reading of other timestamp (of later times) is received then they stored in the queue until the other counterparts arrive. Once all the emotion values are received they are concatenated in a c++ array in the order of their ids and published to the node which compares them. Once the emotion_comparator receives the emotions it simply checks which Turtlebot has best emotion and sends the id of that Turtlebot to all of them.
After this I edited the photo-clicking node which would be launched in the Turtlebot Rapp file on Turtlebot clients. Previously for a single robot system, this photo-clicking node would click photo only if it found the person smiling. But in multi-master setup since we want the best photo, we have to change it account for other robots as well. So I edited the photo-clicking node to subscribe to the photo-comparator node to its topic of Turtlebot_id flag. If a particular photo clicking node on a Turtlebot client receives its own id it would go ahead and click the photo and if it does not receive its own id then it would simply not click.

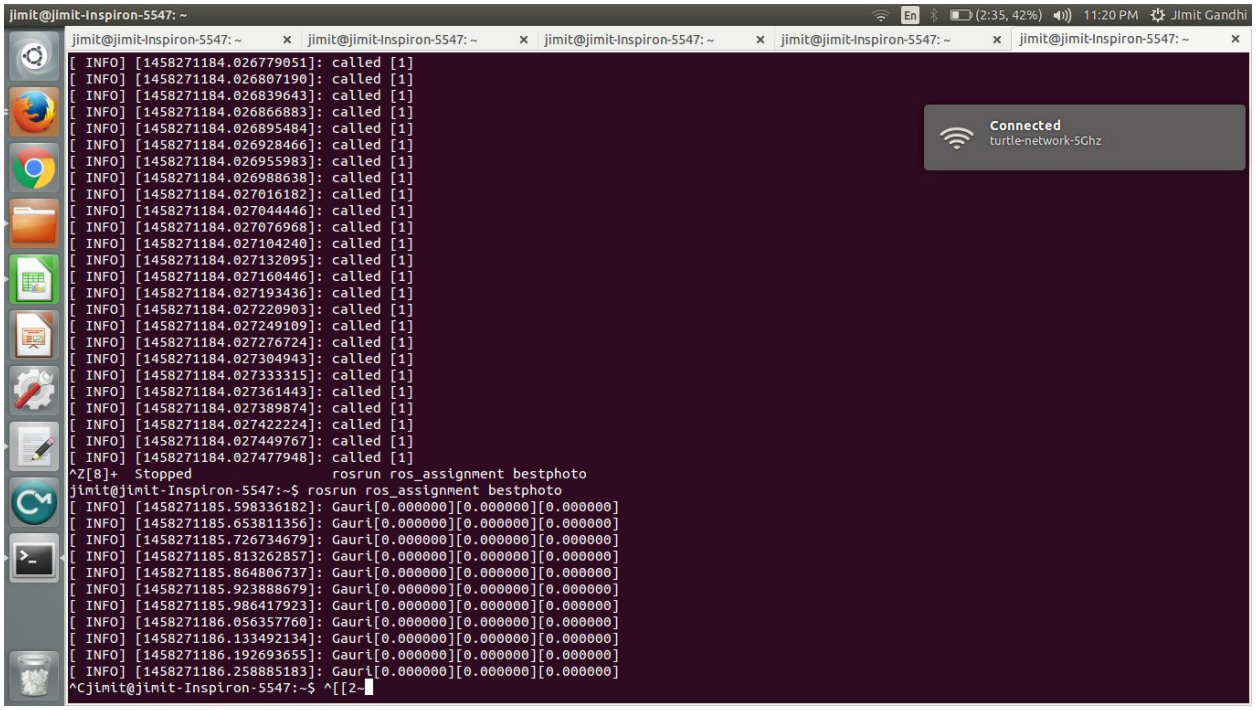


**Figure 1: The "called[1]" above stands for the Turtlebot id which has been called to take the picture. The values below shows the expression values being concatenated and received at the HUB (currently 0 as there is no person smiling)**

Apart from this I also helped Sida tackle the problem of detecting the single person in case of multiple people. Intraface is out of our control and a black box for us. It detects the expression of the person whoever it detects first and not the one that we desire. This was pointed out by our MRSD Director and project course mentor John Dolan in the previous semester and has been on list of issues. So Sida was responsible for this. Since we cannot control Intraface and its working but can control what we input to it I asked Sida to create a node that would subscribe to april tag node to the topics of the tag id and coordinates as well the video frame and crop the entire image except the area above the tag where the head would be. Since we know the person would be at 1 meter approximately from the robot with April tag being on the chest, we know where his face would lie. We also took into account the different heights of people and had a buffer region in the extracted window for these effects as well. This node would then publish its stream to the Intraface node. I also helped Sida tackle with a problem she faced in creation of this node which I will explain in the Challenges Section.

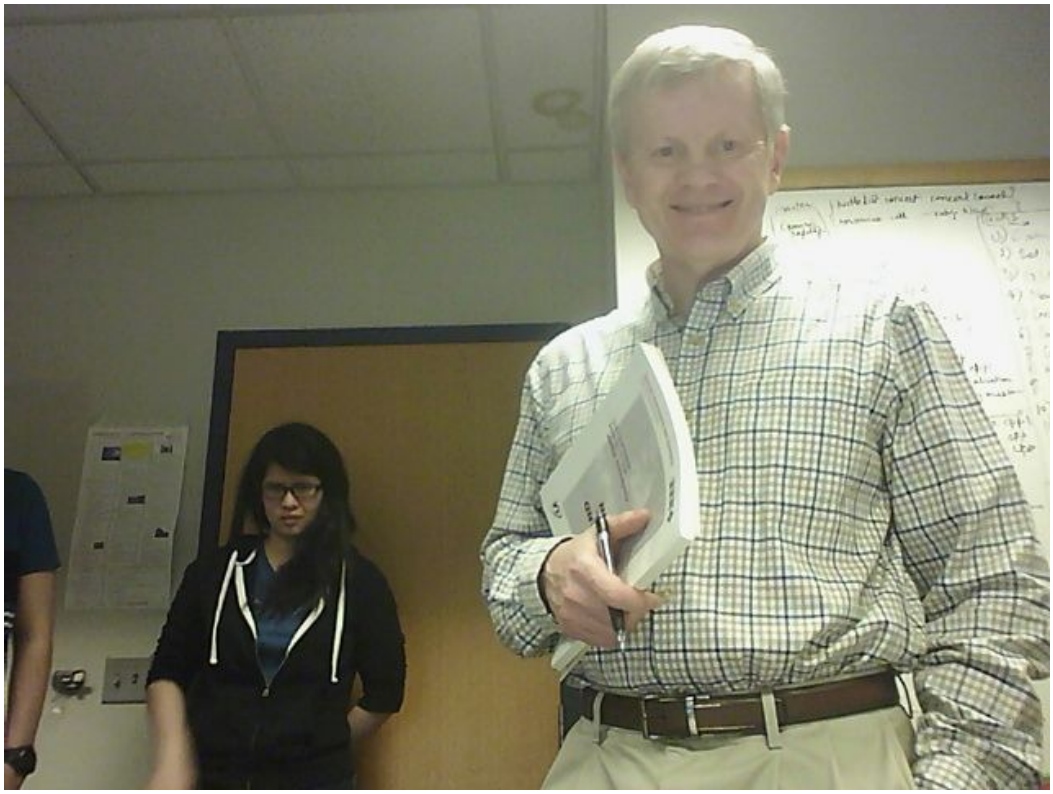

**Figure 3: Result of Progress Review!! Success!!**

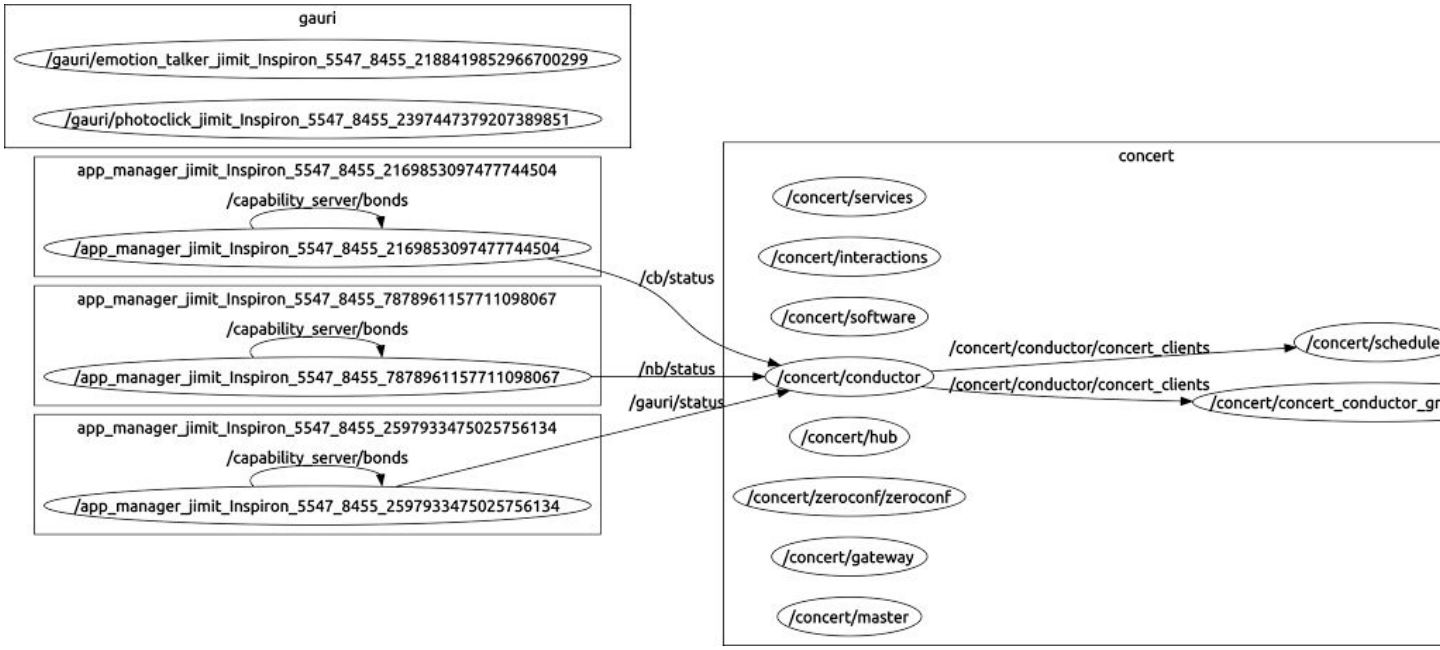


**Figure 2: Concert hub on the right and three other master clients on the left with names /gauri, /cb(chromebook), /nb (netbook) which can be seen on the arrows connecting them**

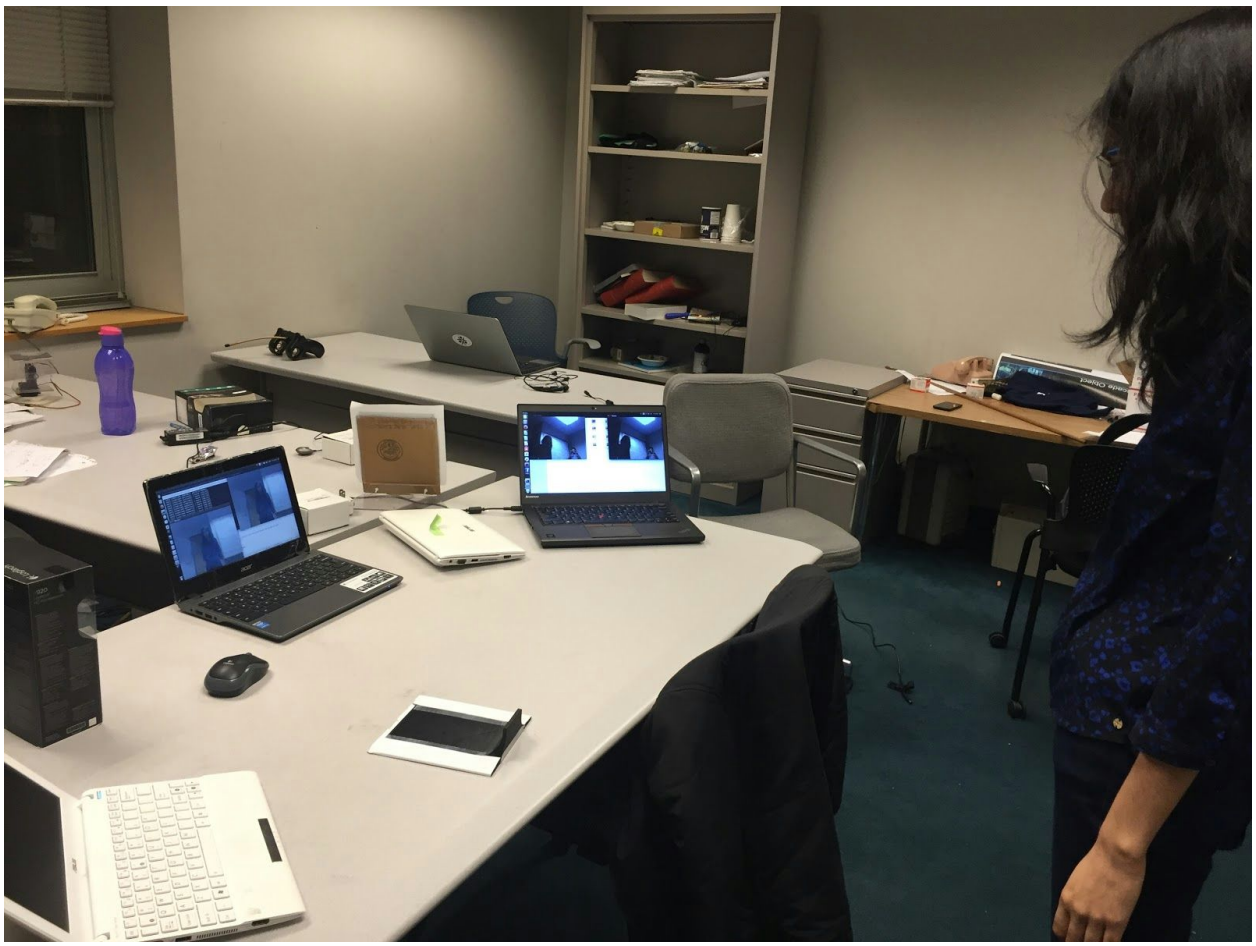

**Figure 4: Gauri standing in front of the three laptops which are analysing her expressions whose values are 0 (figure 1) as she is not smiling**

## Challenges Faced

Both, me and Gauri actually created the Rapps in the last PR9 itself but we were unable to do so since we missed out on the subtle changes such as the one regarding the format of Rapp file I mentioned. One other challenge was the one where we had to mention the list of package dependencies in the .bash config file in a multi-master setup. In a multi-master setup especially when multiple computers or machines are involved, the HUB should know which packages every other master(clients) are  going to launch. If the HUB is unaware of these packages then it simply not recognize and not communicate topics to other masters. So we created this config file with help and advice from the Phd student Sasanka Nagavelli who we report to for the project update in the lab.

Another challenge we faced was get the nodes of the client to correctly subscribe to the topics. Once we had the first successful multi-master launch, we realised that none of the topics were being received by the subscribers though they were being published by the publishers. On probing this problem a little further we realized it was due to change of namespace. For example if a publisher in one master publishes a message on the topic by the name "/emotion_value", the name of the topic in that master's namespace is the same.Since every master has its own local namespace, this topic would appear with a different name on other master. For instance on the HUB this would appear to be as "/$Master-name/emotion_value". So we needed to remap them correctly. We had no idea initially how to map them correctly and not a lots of faqs or issues related to multi-master is posted on internet. So we decided to explore it using intuition and trial by error. So we hard-coded the topic name in the source scripts according to the local namespace for each machines. Obviously we wanted to automate it later on. We first figured out which combination would work. Once that was done and we realised which combination works and formed our hypothesis. Accordingly we automated it and it worked. This can be done in the setup.bash file for each of the Turtlebot. This challenge took quite a few hours to tackle but it was tackled successfully.
The other challenge was the one Sida had faced when she created a node to get desired person's expression. After she created and ran the code for the first time she found the program to be very slow and unstable. Sometimes it would crash. So she asked me to take a look. On inspection I realised that crash occurred due to segmentation fault. Segmentation fault occurs due to reasons of leaked memory or null pointer or when one pointer is trying to access the array element that does not even exist. Turns out, it was due to the third reason. When Sida was extracting the window above the April tag she actually hard-coded the pixel distance. This would fail at the boundaries of the video frame where the window size would exceed the space left to any side of the image. Once she corrected this, the crash problem was solved but the output was still very very slow. Since our chromebooks are new and fast, I was sure that this was not the processor problem. So I decided to take a close look at the code since in my team, I have the highest experience in ROS. After one -two hours of inspection and debugging techniques, I realised that Sida was actually using the April Tag node which Gauri had made in the first semester to do the window extraction. Since there were many functionalities in April tag library that we were using Gauri made an April tag class so she could create multiple instances of April Tag objects. This node was not subscribing to anything during that time. Sida added another functionality of publishing and she did all of that in the class itself. Now since multiple class instances are being created at the same time, the publisher was also being called and along with it many empty node instances were created which is not desirable. We require node to be created only once. Hence it was slow. I removed that part of the code it everything worked out perfectly. The output is as expected.

**TeamWork**

Rohit worked on fabrication of his newly designed and much lighter pan-tilt unit. He fabricated them and also integrated them with motors. The parts fit perfectly and now the pan tilt is aesthetically very good and looks professional works. He did the job even though he faced problems with the 3D printer. Tiffany worked on the fine tuning the pan tilt algorithm she made in

last semester to the new design. It is a work in progress since pan-tilt units were fabricated just few hours before the progress review due to 3D printer problems and large demand of 3D printers. Meanwhile she also focused on navigation part. She researched and studied the flocking algorithm which Allard had implemented last year. She was ready to implement it but one of the global camera stopped working correctly as it was out of focus and blurry. Sida worked on extracting the expression of desired person as I mentioned above. Gauri worked hard and did a terrific job on ROCON. She was the first one to get the ROCON Rapp working right. After that it was just replicating it. She worked side by side with me on client nodes while I was working on the server or the HUB nodes. She also did a great job of revising the number of tasks and steps towards the final spring validation experiments.

**Future Work**

Our multi-master subsystem is almost over. Only few things are left. For the next progress review I plan to focus more on the navigation and finish the flocking part with Tiffany. I also plan to finish the part where robots arrange themselves around the person at -45, 0 and 45 degrees with Gauri. I also plan to test the detection subsystem once Sida finishes integrating her node with our multimaster framework.