The Robographer: Progress Review #12

Name: Rohit Dashrathi
Team: G (The Robographers)
Teammates:
Jimit Gandhi
Gauri Gandhi
Rohit Dashrathi
Sida Wang
Tiffany May
ILR No.: # 11
Submission Date: April 13, 2016

1. Individual Progress

Responsibilities: Project management
Mechanical Design and Development

Softwares/tools Used: Google Drive, Google Calendar, Microsoft Excel, Solidworks, Makerbot Desktop

Task Description:

Following tasks were assigned to me and completed before the progress review 12:

a. Conducting team meeting and deciding goals for PR#11
b. Correcting the face tracking algorithm
c. Implementing the face tracking over all 3 turtlebots

a. Setting the goals for PR#12:

1. Successful localization of the turtlebots
2. Collaborative navigation of turtlebots
3. Face tracking using pan tilt unit

b. Correcting the face tracking algorithm:

This was the time for some change of role for me. I have completed the mechanical work for our project. Hence, I decided to undertake another area of pain. We were struggling to get the face tracking using the pan tilt camera units since last 3 progress reviews. Though Sida and Tiffany were trying to interface the PTZ units with face tracking, they have not been successful till now. We had a working face tracking code for a single camera which we demonstrated in the Fall validation experiment. However, this code was configured only for one particular camera model and had no provision so as to suit the different camera versions. Tiffany and Sida had made some changes in the same to operate the new camera units we bought. However, they had made edits in the old working code only without saving it. As a result, we lost the working code. Now the task was to rectify the code to achieve face tracking.

I decided to take it up upon myself by approaching the problem from basics. I was not involved during the program generation for face tracking. For this, I studied the code thoroughly. I noticed that a lot of hardcoding was done over the previous working code in the process of new code generation. I removed this hardcoding, re-defined the code logic and finally managed to retrace the primary version of the code, which Sida and Tiffany had developed with the help of our advisor Sansaka Nagavelli.

Finally, I noticed that the main function which operates the face tracking was called as proportionalController (). This function used the input values published by the ROS node written to operate our Robographer system on the Arduino IDE serial monitor. These input values were a set of (x, y) co-ordinates of the nose position of the detected person. These (x, y) co-ordinates were fed to the proportional controller function. This function then interpolates these input values and converts to 2 angular values. This angle values were then fed to the pan and tilt motors respectively.

The interpolation equation in the proportional controller function had 2 gain parameters, gainX and gainY (many thanks to Sasanka ☺). These gains were provisioned so as to cope up with the different types/makes of cameras. I played with these gains while connecting to our newly bought camera (Logitech 920c) and finally achieved the desired face tracking motion. Though I have not done much of the electronics/CS work previously, I enjoyed doing this work and finally achieved success.

c. Checking the face tracking:
Even though the camera moments were smoother than before while performing the face tracking operation, the movement angle was still a problem. The camera on turtlebot 1 detected the face but the pan motor went in the opposite direction after detection. This meant that the angle fed to the panning servo was in the opposite direction. However, the servos can't be fed a negative angle. Hence, I subtracted the output pan angle from 180 degree and the servos worked fine.

d. Implementing the face tracking:
The pan tilt unit on the Turtlebot 1 was now configured. However, I still needed to configure the pan tilt units on the turtlebots 2 and 3. Due to the inconsistent mounting of servo headers, there was a possibility of error in the face tracking angles fed to the servo motors. Hence, all 3 pan tilt units needed corrections in their output. I made the necessary connections and got all 3 pan tilt units operated.

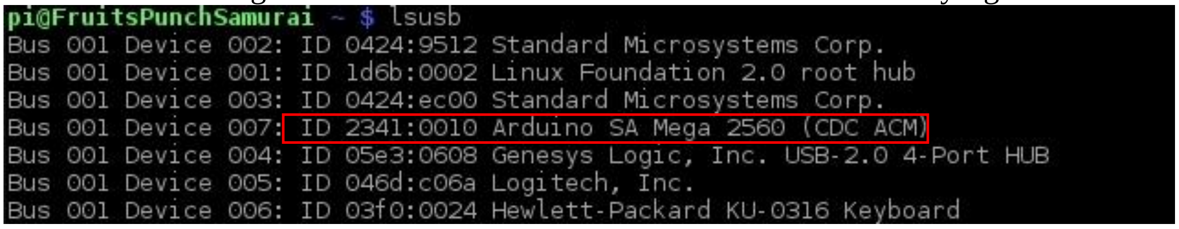
2. Challenges

The PR#12 was full of challenges. The completion involved some major challenges as described below:

1. **'The algorithm':**
   This was the most intriguing challenge. Usually an algorithm is developed from zero to a correct version. Here I had to trace back to the zero version from an incorrect algorithm. This took a great amount of time.

2. **Instability of the Arduino port on Ubuntu:**
   Running the Arduino IDE on Ubuntu has a weird problem of serial port instability. Running Arduino is windows smooth and windows assigns a permanent port to Arduino microcontroller as per the USB port to which it is connected. However this does not happen in Ubuntu and the address of Arduino port keeps changing. With some research I was able to encounter this issue using the following procedure to assign persistent name to the USB hardware:

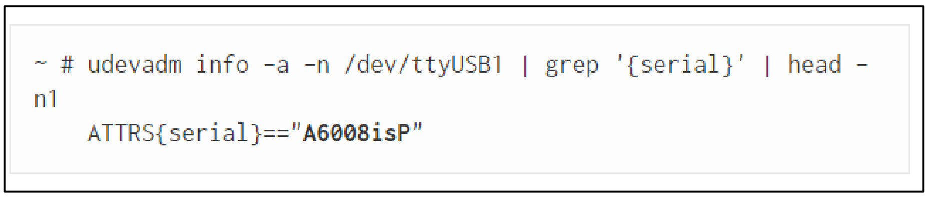
   a. Extracting the vendor ID and product ID attributes of Arduino device:
   This was done using the $lsusb command in Ubuntu terminal as shown by figure 1:



```
pi@FruitsPunchSamurai ~ $ lsusb
Bus 001 Device 002: ID 0424:9512 Standard Microsystems Corp.
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 001 Device 003: ID 0424:ec00 Standard Microsystems Corp.
Bus 001 Device 007: ID 2341:0010 Arduino SA Mega 2560 (CDC ACM)
Bus 001 Device 004: ID 05e3:0608 Genesys Logic, Inc. USB-2.0 4-Port HUB
Bus 001 Device 005: ID 046d:c06a Logitech, Inc.
Bus 001 Device 006: ID 03f0:0024 Hewlett-Packard KU-0316 Keyboard
```

Figure 1: Extracting the vendor ID and product ID of the connected Arduino (representational figure)

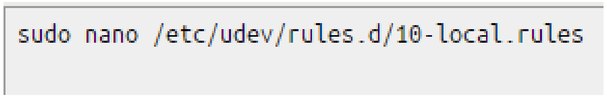
b. Extracting the serial number attribute:
This was done using the ‘udevadm’ command in the Ubuntu terminal as shown in the following figure:



```
~ # udevadm info -a -n /dev/ttyUSB1 | grep '{serial}' | head -
n1
    ATTRS{serial}=="A6008isP"
```

Figure 2: Extracting the serial number of the connected Arduino (Representational Figure)

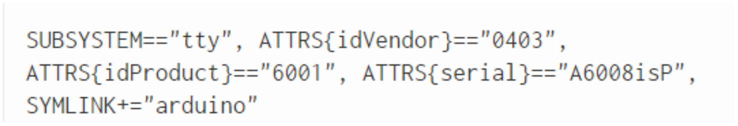
c. Writing the UDEV rule for the connected Arduino:
The UDEV rule configures the Arduino to a permanent symbolic link. The UDEV rule is nothing but a system administered text document which enables the read/write privileges to a particular hardware/software. Although being a text file, it cannot be directly created using the Ubuntu system interface and has to be created through the Ubuntu terminal. The UDEV rules are located in lib/udev/rules.d/<*rule-name*> or etc/udev/rules.d/<*rule-name*>. They are written as follows:



```
sudo nano /etc/udev/rules.d/10-local.rules
```

Figure 3: Writing UDEV rule in Ubuntu terminal for Arduino (representational figure)

This opens the text file in ‘nano’ text editor. Any other text editor can also be used by replacing ‘nano’ in above command by the respective terminal name. The rule is further written in the opened text file where the key distinct attributes such as the productID, vendorID and the serial number (obtained as in figure 1 and 2) are put:



```
SUBSYSTEM=="tty", ATTRS{idVendor}=="0403",
ATTRS{idProduct}=="6001", ATTRS{serial}=="A6008isP",
SYMLINK+="arduino"
```

Figure 4: Writing the DEV rule (representational figure)

This configures the file *10-local.rules* as a developer rule (DEV rule) for the Arduino and directs the Arduino to a permanent port address such as *ttyACM0* etc.

3. Teamwork

Since the start of the project, I have taken the responsibility for the completion of the mechanical work subsystem. With the work done before PR#11, the mechanical subsystem now stands completed totally for the project. With the work done before PR#12, I also undertook the work of Arduino programming. Jimit and Gauri once again worked relentlessly to complete the localization and navigation part. They also did well to fix the repetitive voice command issue. Even though they had some consistent problems in navigating the bots, they managed to complete the localization module successfully. Tiffany and Sida worked on AprilTag tracking using the pan tilt cameras. However, she had a few problems to implement the same for all the three units. With this PR, we were able to complete most of our subsystem work this time as well as their integration to an extent. However despite having better preparations, we had problems during the PR#. Hence, our overall presentation for the PR#12 did not go very well.

4. Future Plans

My individual plans for the SVE for the Robographer Project:

1. **Complete integration of Arduino with ROS using rosserial:**
   The system now has an ability of face tracking. However, every time after starting the system, the face tracking code needs to be burned on the Arduino bootloader. I am now working to access the Arduino using rosserial feature which will eliminate this issue.

2. **The final touch:**
   The fabrication of the mechanical part of the system is now complete with 3 turtlebots, fitted with the pan tilt camera units. However, a permanent arrangement for some important components such as the Arduino Microcontroller, the AprilTags for localization along with the wiring arrangements still needs to be done. I have planned to complete this work before the PR#11 and will make sure to present the integrated Robographer system with an aesthetic look of a finished product. Moreover, I have planned to conduct the complete integrated system testing in order to ensure safe and guaranteed demonstration during the SVE and SVE encore.

5. Appendix

Assigning persistent addresses to serial devices:

http://hintshop.ludvig.co.nz/show/persistent-names-usb-serial-devices/

http://ubuntuforums.org/showthread.php?t=168221