Progress Review 1

Tushar Agrawal

Team A - Avengers
Teammates: Adam Yabroudi, Pratik Chatrath, Sean Bryan

ILR #2

October 23, 2015

1. Individual Progress

For the week, I was responsible for setting up the BeagleBoard xM with an Ubuntu 14.04 (LTS) image with a working internet connection, ssh server, ROS and hopefully OpenCV.

Ubuntu 14.04

The preconfigured image was a 13.04 ubuntu console image with ROS set up.

Firstly, I saved a backup of the 13.04 image from the SD card. Then, I downloaded the latest pre-configured Ubuntu 14.04 image from the internet and set it up on the micro SD card [1]. When I tried to boot up the beagleboard using that image, I was not able to see any data on the screen connected to the beagleboard (using HDMI). Worried it might be due to a pre-configured image, I started setting up the image builder platform for configuring beagleboard images on my laptop.

After realizing that several other configuration settings are required to get the screen working, I figured it would be better to verify the boot-up sequence over the serial console.I got a USB to RS-232 converter and checked the console output when the board tried to boot up (at 115200 baud and 8N1 port settings). The boot-up sequence was running into an error while starting up the kernel from the SD card.

```
mmc0: error -110 whilst initialising SD card
```

Searching further, there was a possibility of a wrongly partitioned SD card or a faulty one.

I tried different ways of formatting the SD card without success. Finally, I took a spare 8GB SD card from Adam and tried to install the same image onto it. It worked instantly. I confirmed that all of the configuration settings of the pre-configured image were good enough for our requirements, I started installing required packages onto that.

Internet, SSH

I confirmed that the Beagleboard could connect to the internet using the Ethernet to home network, I quickly installed openssh-server to access the board over the network instead of the console.

ROS

Then, I started installing ROS on Beagleboard [2]. Once done, I tested the ROS environment by hooking up the Hakuyo LIDAR unit (that Adam was trying to test with ROS over Rasp Pi), and copying their ROS node into the beagleboard. We could immediately see the ranges from the Hakuyo sensor for all angles as messages over a ROS topic (/scan).

Camera

Excited with the progress, I plugged in the Logitech camera and used fswebcam to click an image from it and save it. (I could not view it on the beagleboard as there was no GUI on it). I transferred the same to my computer and was satisfied that the camera was working well with the Ubuntu image (Image 1).

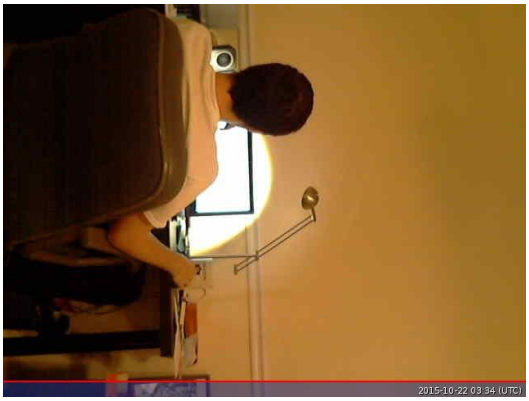

Image 1. *Image captured on the Beagleboard and transferred. (Adam working)*

OpenCV

On hearing Adam's concerns about problems installing OpenCV with ROS, I realized it would be best to first try a clean install of OpenCV and try using it with ROS on my laptop. And only after that try installing OpenCV on the beagleboard. I went ahead and installed the 2.4.11 version of OpenCV on the laptop and got it working in the first try, eventhough it took a considerable amount of time (as it got built from its source). I tested using it with ROS by making a ROS node which used OpenCV to draw a tiny circle on the camera stream and published the modified stream [3]. The stream could then be subscribed by image_view and the modified results could be seen (Image 2). (Code is attached in appendix-A)

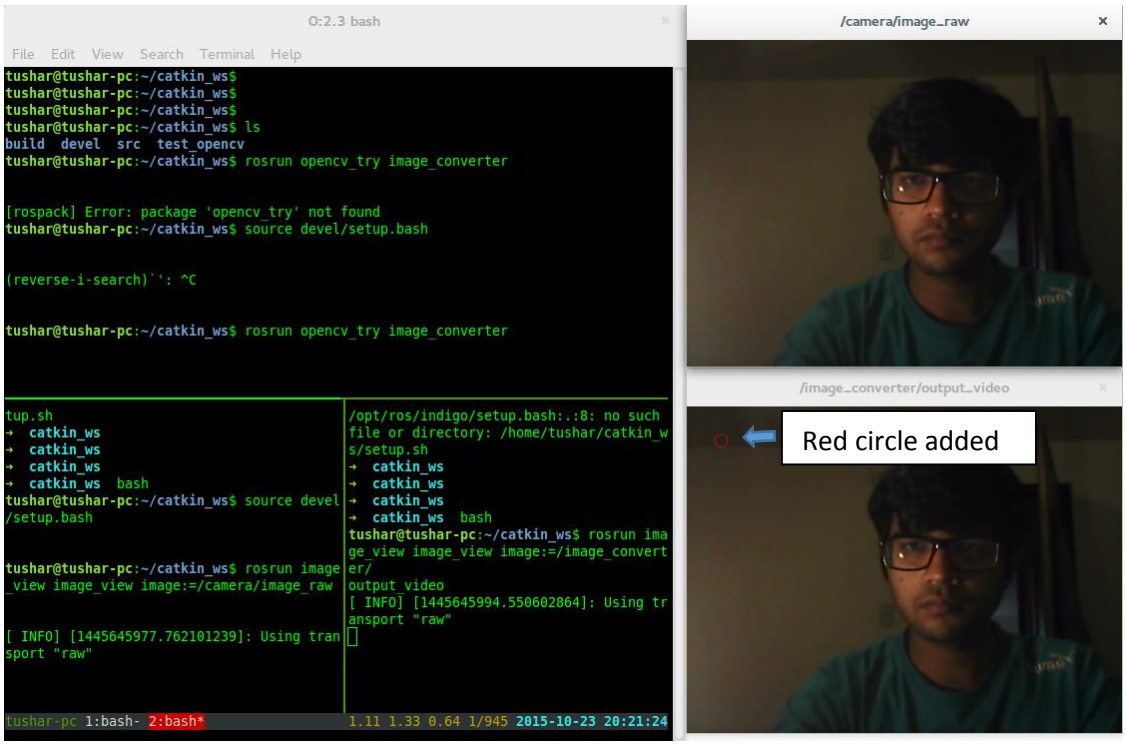

Image 2.OpenCV with ROS - Adding red circle to camera stream (On Laptop).

Next steps include setting up OpenCV on the beagleboard and testing it with ROS. Please refer image 3 for the current system setup.

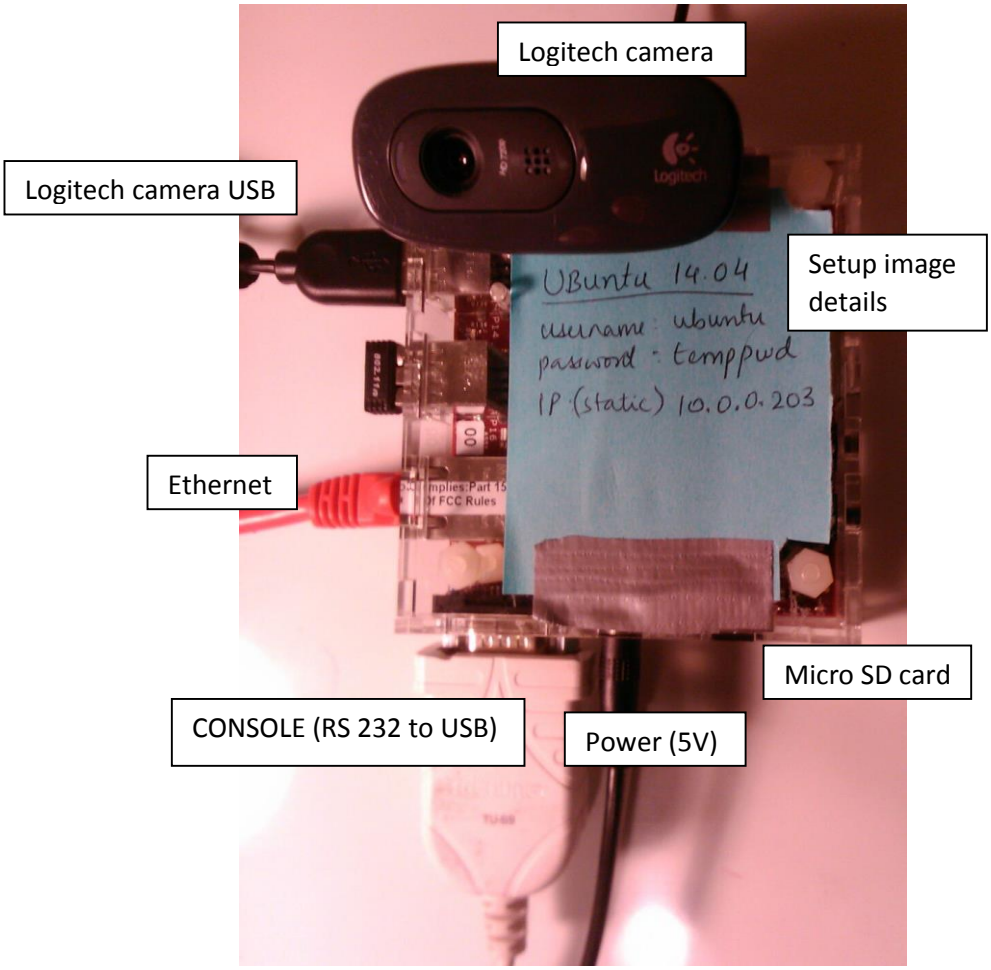

Image 3: BeagleBoard setup. Including peripherals and connections

2. Challenges

I faced most challenges while debugging the first issue in the new 14.04 image on the SD card used at first. After trying different images, and formatting options the mmc error -110 was not going away. Only after replacing the SD card, I could get it to work.

I also faced some challenges when making the IP static for the Beagleboard, to be able to ssh easily onto it. For some reason, internet stopped working on the beagle board. Finally, I let dhcp obtain the IP address and simply looked it up through the console after every reboot. I shall be fixing that this week.

I faced another issue while trying to click a picture from the webcam using the command line. Several tools were available for the same like mplayer, streamer, cheese, etc. But many of those had X server dependencies and wanted to avoid installing X server to keep the

beagleboard image light. I was able to find the fswebcam tool that was a small nifty tool for command line webcam control.

3. Teamwork

We split up the immediate goals of the project amongst the four team members. I was responsible for setting up the Beagleboard with an appropriate Ubuntu image, ROS and if possible, OpenCV and work closely with Adam in comparing its performance with that of Raspberry Pi.
Pratik was responsible for working on the obstacle avoidance system by testing the available proximity sensors for their ranges, cones and other configurations. Sean was working on getting a basic model available to Pratik to be able to mount sensors and test. He also took up the task for setting up the website and coordinating with Birds Eye View Aerobotics for possible technical and material help with the UAV, the Firefly6.

I coordinated with Adam to see the pros and cons of using the Beagleboard vs the Raspberry Pi. We realized that the Raspberry Pi was very slow compared to the beagleboard (especially after installing ROS on it) and much harder to set up. We also set up the Hakuyo ROS node to test the Hakuyo and ROS on the beagleboard.

4. Future Plans

In this week, Pratik is going to work with Sean to speed up the analysis of the obstacle avoidance system. They will try to get specific metrics about the sensors' individual performance and their combined performance in different environments. We hope to be able to categorize IRs and Ultrasonics on the basis of whether they can contribute to obstacle avoidance in our application or not.

Sean will be continuing his conversation with the Birds Eye View Aerobotics and hoping to get valuable help in terms of acquiring a vehicle or technical knowledge.

Adam will be setting up the Pixhawk board and interfacing it with a motor. He will also work on getting the NicaDrone (electro-permanent magnet) which will be our package delivery mechanism. As Nicadrone is out of stock, but its BOM and board layouts are open source, Adam might have to get the board printed and order the parts separately and finally, assemble it later.

I will continue setting up the beagleboard with OpenCV and develop basic algorithms for marker detection using a camera on the Beagleboard. The first aim to be able to see what FPS we are able to achieve with a basic algorithm. Then, in the week after that can select specific markers.

References

[1] Ubuntu on Beagleboard: http://elinux.org/BeagleBoardUbuntu
[2] ROS on Ubuntu http://wiki.ros.org/indigo/Installation/UbuntuARM
[3] ROS OpenCV test code
http://wiki.ros.org/cv_bridge/Tutorials/UsingCvBridgeToConvertBetweenROSImagesAndOpenCVImages

Appendix A – OpenCV ros conversion and draw a circle on output image ([3])

```
#include <ros/ros.h>
#include <image_transport/image_transport.h>
#include <cv_bridge/cv_bridge.h>
#include <sensor_msgs/image_encodings.h>
#include <opencv2/imgproc/imgproc.hpp>
#include <opencv2/highgui/highgui.hpp>

static const std::string OPENCV_WINDOW = "Image window";

class ImageConverter
{
  ros::NodeHandle nh_;
  image_transport::ImageTransport it_;
  image_transport::Subscriber image_sub_;
  image_transport::Publisher image_pub_;

public:
  ImageConverter()
    : it_(nh_)
  {
    // Subscrive to input video feed and publish output video feed
    image_sub_ = it_.subscribe("/camera/image_raw", 1,
      &ImageConverter::imageCb, this);
    image_pub_ = it_.advertise("/image_converter/output_video", 1);

    cv::namedWindow(OPENCV_WINDOW);
  }

  ~ImageConverter()
  {
    cv::destroyWindow(OPENCV_WINDOW);
  }

  void imageCb(const sensor_msgs::ImageConstPtr& msg)
  {
    cv_bridge::CvImagePtr cv_ptr;
    try
```

```
    {
      cv_ptr = cv_bridge::toCvCopy(msg, sensor_msgs::image_encodings::BGR8);
    }
    catch (cv_bridge::Exception& e)
    {
      ROS_ERROR("cv_bridge exception: %s", e.what());
      return;
    }

    // Draw an example circle on the video stream
    if (cv_ptr->image.rows > 60 && cv_ptr->image.cols > 60)
      cv::circle(cv_ptr->image, cv::Point(50, 50), 10, CV_RGB(255,0,0));

    // Update GUI Window
    cv::imshow(OPENCV_WINDOW, cv_ptr->image);
    cv::waitKey(3);

    // Output modified video stream
    image_pub_.publish(cv_ptr->toImageMsg());
  }
};

int main(int argc, char** argv)
{
  ros::init(argc, argv, "image_converter");
  ImageConverter ic;
  ros::spin();
  return 0;
}
```